AFIT/GST/OS/84M-5

AD-A141 092

BRIK
AN INTERACTIVE, GOAL PROGRAMMING MODEL FOR NUCLEAR EXCHANGE PROBLEMS

THESIS

ROBERT E. BUNNELL, CAPT, USAF
RICHARD A. TAKACS, CAPT, USAF

AFIT/GST/OS/84M-5

DTIC FILE COPY

Reproduced From
Best Available Copy

DTIC
ELECTE
MAY 16 1984
S D
E

Approved for public release; distribution unlimited

20000803068

84 05 15 024

AFIT/GST/OS/84M-5

Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

BRIK

An

Interactive

Goal Programming Model

For

Nuclear Exchange Problems

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science

by

Robert E. Bunnell
Capt USAF

Richard A. Takacs
Capt USAF

Graduate Strategic and Tactical Sciences

March 1984

Approved for public release; distribution unlimited

PREFACE

BRIK is an interactive nuclear exchange model which uses preemptive goal programming for the allocation process.

We wish to thank our co-advisors, LtCol Ivy D. Cook and Major James K. Feldman for their continuing direction and invaluable insight. LtCol Cook's patience and Maj Feldman's initial suggestions concerning various model options proved invaluable. Both allowed us to get off to a good start. We also want to show our appreciation to the Association of Computing Machinery for allowing us to use their partitioning algorithm for goal programming, PAGP, as the allocation algorithm in BRIK.

Finally, we especially wish to thank our wives and families, who through their understanding and support, encouraged the rapid completion of this thesis.

Robert E. Bunnell

and

Richard A. Takacs

Contents

List of Figures

List of Tables

ABSTRACT

BRIK is a unique nuclear exchange model (NEM). It is the first NEM to be both totally interactive and to rely entirely upon preemptive goal programming to drive its allocations. The program is written in Fortran 77 and exists on the VAX 11-780 using the UNIX operating system. Since it includes an internal allocation subroutine, it is therefore completely transportable.

BRIK will allocate weapons to targets in order to meet damage expectancy (DE) goals of the user. The decision variable is a non-integer geometric mean representing the number of weapons allocated to an entire target class, using any of three objective functions and up to thirteen different allocation rules. The program is useful for a wide range of scenarios including meeting DE goals with the available arsenal or creating an arsenal to meet established goals.

The programmed model uses a preemptive linear goal programming algorithm, allowing explicit preferential treatment of target classes, which can be placed into any of seven priorities. Target values can be input to differentiate target classes within the same priority. BRIK computes single shot probability of survival using the nuclear targeting vulnerability (NUN) system of the Defense Intelligence Agency or from vulnerability expressed as a PSI overpressure number. As currently dimensioned, the program can handle up to 20 target classes, 23 weapon classes, and 20 hedging constraints.

BRIK is a unique nuclear exchange model (NEM). It is the first NEM to be both totally interactive and to rely entirely upon preemptive goal programming to drive its allocations. The program is written in FORTRAN 77, and exists on the VAX 11-780, using the UNIX operating system. It includes an internal allocation subroutine, and is therefore completely transportable.

BRIK will allocate weapons to targets in order to meet damage expectancy (DE) goals of the user. The decision variable is a non-integer geometric mean representing the number of weapons allocated to an entire target class, using any of three objective functions and up to thirteen different allocation rules. The program is useful for a wide range of scenarios, including meeting DE goals with the available arsenal or creating an arsenal to meet established goals.

The programmed model uses a preemptive linear goal programming algorithm, allowing explicit preferential treatment of target classes, which can be placed into any of seven priorities. Target values can be input to differentiate target classes within the same priority. BRIK computes single-shot probability of survival using the nuclear targeting vulnerability (VN) system of the Defense Intelligence Agency or from vulnerability expressed as a PSI overpressure number. As currently dimensioned, the program can handle up to 20 target classes, 20 weapon classes, and 20 hedging constraints.

The report describes the physical and mathematical model, scenario limitations, input requirements, and the damage function. It includes a user guide, variable and subroutine definitions, and a computer listing.

I Introduction

The existence of nuclear arsenals has progressed from the production of the first atomic bomb to the development of enhanced "traditional" triad elements such as the B-1B bomber, the MX missile, the SS-18, and the Backfire bomber. This activity has resulted in the deployment of a substantial number of delivery systems and nuclear weapons. Also, large expenditures of funds on research and development have produced systems which for one reason or another never reached deployment.

When nuclear arsenals began to grow, analysts attempted to determine the "Nuclear Balance." In the early 1950's, the US used straight inventory comparisons (Ref 13). With time, these static measurements proved inadequate and failed to give a true indication of a force's capability.

When Robert McNamara became Secretary of Defense in 1961, he introduced "Defense Economics" (Ref 3:1.5). The ingredients of this dynamic approach to nuclear force planning required (1) identifying the force's mission objectives and (2) evaluating how well alternative forces could meet these objectives. Thus, force assessment techniques changed from static to dynamic measurement.

Today, analysts and policy makers are still using a dynamic approach to assess the adequacy of the US strategic arsenal to deter the Soviets and to assess the effect of new US and Soviet weapon systems on the strategic balance. To

assist analysis, a number of nuclear exchange models (NEMs) have been built.

These models generally fall into two broad categories -- computer simulation models and analytical models. Computer simulation models are used to establish targeting strategies for various contingencies (Ref 27:1). These models consider many variables. Consequently, they require many hours of computer time and are generally inflexible regarding varying applications. Unlike simulation models, analytical models use expected value concepts. They are used within the DOD to study the issue of optimizing weapons allocation.

Analytical allocation models found in the literature differ considerably in degrees of realism and flexibility. Realism refers to the model's output reflecting an actual representation of what is possible. An allocation model is realistic if the assignment of a weapon to a target is in fact feasible. An example of an infeasible assignment is an allocation of a weapon to a target that is beyond the weapon's range. Flexibility refers to the model's ability to handle many different types of strategic allocation problems. This is achieved by allowing a variety of allocation rules and objectives.

Problem Statement

A review of the analytical allocation models indicated two things. First, if they have the flexibility to handle many different scenarios, they are large and their design

favors batching, thus prohibiting interactive analysis. Second, if they are of reasonable size to permit interactive analytics, they are limited in scope and capabilities.

Objective and Scope

The objective of this research effort was to build an allocation model that was both realistic and flexible, yet of reasonable size to permit interactive analytics. Milan Zelany says "...researchers seem to be evolving a consensus: let the human being, human decision maker, become a core around which to build our techniques, adapting them to his needs and amplifying his own decision making powers" (Ref 28:2). Nuclear exchange models, with their batch mode designs, prohibit any convenient human interaction with the computer. Therefore, it was the intent of this effort to develop a model that provided the qualities of the larger models yet allowed interaction between the analyst and the machine.

A model of this nature should not be restricted to a single capability. Therefore, building this model, BRIK, to perform just one type of analysis would have defeated the purpose of this effort. An analyst who would potentially use an NEM should have the ability to perform a variety of tasks. Therefore, the capability of this model exceeds the requirements of any one single analysis. Given a weapon base and a target base, the model performs an allocation based upon a variety of objectives. For example, BRIK includes the following:

1) It optimally allocates weapons to meet targeting goals.

2) If goals are under-determined, i.e., if the goals are met and there are weapons remaining, the model could minimize the number of warheads, megatonnage, equivalent megatonnage (EMT), or countermilitary potential (CMP).

3) It determines what force is necessary to meet damage expectancy (DE) goals.

To install realism in the allocation process, numerous allocation rules are allowed. For example, the model permits the following:

1) Designation of inappropriate weapon/target assignments.

2) Attaining minimum levels of DE on designated target classes.

3) Attaining minimum levels of DE on designated target classes using a specific set of weapons.

4) Enforcing an upper level of damage on designated target classes resulting from a specific set of weapons.

5) Restricting the number of a class of weapons which can be allocated to a designated target class.

6) Restricting the number of weapons which can be allocated to each target in a particular target class.

7) Building any user-defined allocation rule.

To increase the interactive capabilities of the model, BRIK allows the user to ask the following "what if" type

questions while seated at a terminal:

1) What if the number of available weapons changed?

2) What if the DE goals changed?

3) What if weapon characteristics changed?

4) What if target characteristics changed?

This concludes the description of the objective and scope of this thesis. The next section describes the approach followed by the authors in producing this model.

Approach

This effort was a continuation of a thesis completed in December 1982, by Michael C. Wambsganss (Ref 27). He demonstrated that auxiliary variables in linear programming constraints could be used to allocate weapons to targets in order to meet user-defined damage expectancy goals. His methodology provides the foundation for this effort and will be reviewed in the literature review section.

After studying Wambsganss' work, it was necessary to design BRIK. To increase transportability, the authors decided that BRIK should not use a library package for its allocation routine. Therefore, a search was conducted for a suitable linear programming package. The authors finally decided to use PAGP, a goal programming algorithm acquired from The Association for Computing Machinery. Next, time was spent determining what allocation rules should be included and how the objective functions should be designed. With the basic model adequately outlined, the coding stage began.

As BRIK was being written, its design followed an evolution process. While its basic outline is unchanged, some notable features were included well into the project. The two most noteworthy additions are the VNTK vulnerability system and the file systems for the weapon and target bases. The addition of the VNTK system proved to be invaluable as it permitted direct comparison with the Arsenal Exchange Model (AEM) during the final portion of this effort.

Format

Chapter II provides a description of the elements found in most nuclear exchange models: target base, weapon base, and damage function. This is followed by a review of many of the nuclear exchange models found in the literature.

Chapter III presents some characteristics of BRIK. These include model restrictions, scenario limitations, and input requirements. Also, in this chapter is an explanation of the single shot probability of survival computations and a discussion of the damage function.

Chapter IV derives all of the constraints used by BRIK in the linear programming matrix.

Chapter V reviews Goal Programming. This is followed by a description of the mathematical formulation of BRIK, including the objective functions and the matrices. Finally, the assignment algorithm, PAGP, is reviewed.

Chapter VI concerns both verification and demonstration. After a discussion concerning BRIK's "correctness", a

comparison is made between the solutions from a sample problem as computed by BRIK and AEM. This comparison is followed by numerical examples demonstrating BRIK's versatility.

Chapter VII provides a summary of the modeling process, descriptions of potential uses for BRIK, and a discussion of potential improvements to the model.

II Literature Review

The term "model" is used broadly in the literature with various definitions. In this study, a model is defined as a computer program which converts information about the targets, weapons, and employment plans of two sides into a prediction of the outcome of a war. This is commonly called an arsenal exchange. For those readers who are not familiar with nuclear exchange models, a general overview of the model components and terminology follows along with a discussion of several models.

MODEL ELEMENTS

Nuclear exchange models can be grouped into two general categories: computer simulation models and analytical models (Ref 27:1).

Simulation Models. Computer simulation models are used to develop targeting strategies. These models require extensive data preparation, contain a large number of variables, and often use hours of computer time. The complex structure of the model frequently precludes sensitivity analysis because of the large amount of runs necessary to obtain an expected value which has a high degree of certainty. Simulation models are also regarded as generally inflexible and oriented to a specific application (Ref 27:1-2).

Analytical Models. Analytical models center on expected values such as an expected damage level and expected surviving arsenal. Data preparation times, computer run times, and levels of complexity vary widely, increasing in proportion to the level of realism required by the specific model. Analytical models also permit sensitivity analysis of variables.

The analytical model generally attempts to optimize either a weapon allocation or cost. To optimize weapon allocation, the model searches for the highest level of expected damage to the opponent from a specified weapon arsenal. To optimize cost, the model seeks the least expensive combination of weapons which causes a specified level of damage to the opponent. Analytical models are the most widely used nuclear exchange models (Ref 27:2).

Although models differ widely in construction and application, both simulation and analytical models share common submodels: the weapon complex, the target complex, the engagement and allocation rules, the damage function, and the algorithm or solution technique. The weapon and target submodels generally determine the complexity of the other submodels (Ref 27:9). Since model complexity may be an issue to prospective model users, the content of the weapon and target complex submodels is given in more detail, followed by a discussion of terminology that is used in nuclear exchange literature.

Weapon Complex. The weapon complex addresses three distinct categories (Ref 27:10): the scope, the number of weapon system types and their associated penetration aids; weapons reach, which refers to the ability of a weapon to reach a specific target; and commitment policy of each weapon, which refers to the amount of strikes launched, quality of the intelligence and bomb damage assessment, and weapon availability uncertainties.

a) Scope: The scope may specify single or multiple weapon types based on their weapon characteristics such as circular error probable (CEP), the weapon's explosive yield, weapon system reliability, and defense penetration aids. CEP is the radius of an imaginary circle centered on the target in which at least 50% of the weapons aimed are expected to land. The weapon's explosive yield is the size of the weapon usually measured in kiloton or megaton equivalent TNT yields. Weapon system reliability (WSR) is the probability that once the weapon system is launched it will function properly during its mission. Penetration aids are those items that help a weapon system defeat a defen ive system. Chaff and decoys are examples of penetration aids. In many models, a probability of penetration (PTP) simulates enemy defenses. Sometimes, in order to simplify the input requirements, models will combine PTP and WSR into a single probability. Models either assume that all weapons are available (deterministic) or that there

is a certain amount of system unreliability in launch (probabilistic).

b) Weapon Reach: Weapon reach may limit feasible assignments of weapons to targets or degrade weapon accuracy as the range traveled increases. A 0-1 matrix can normally be used to represent weapon-target assignments with a 1 meaning a feasible assignment. The targets that a MIRV (multiple independently targeted re-entry vehicles) can attack are not only limited by the range of the carrier but also by the MIRV footprint. Most models do not consider the MIRV footprint. Instead, all weapons are considered independent. This assumption may provide allocations that are beyond the range of the actual weapon capability.

Range is not the only factor considered in weapon reach. The BSWADE model uses time as a factor to limit feasible assignments. If a target is time urgent, meaning that the target must be hit by weapons that can strike the target within a specified time limit, an allocation will be allowed only from those weapons that can arrive at the target in the specified time period.

c) Weapon Commitment Strategies: Models usually assume that the weapons will be allocated against existing targets. For example, only those silos that contain missiles will be attacked. This aspect indirectly assumes that the attacker has prior intelligence about his adversary's battle plan. Even though most assume some prior knowledge about an

adversary, most models do not consider any battle damage assessment (BDA). The models will continue to allocate weapons against a target until its damage expectancy has been attained. There is no checking procedure to indicate whether a target has been destroyed by an earlier arriving warhead, so that the next warhead can be allocated against a target that has not been attacked or destroyed.

A model can also be categorized on the basis of how many strikes it can compute. Models limited to a single strike are usually limited to evaluating a single goal by a single attacker. A two-strike model can compute the results of an initial attack followed by a retaliatory strike without making two runs. Models that can handle three or more strikes can evaluate the optimum reserve force that could be held back from a strike to discourage an opponent from a retaliatory strike.

Target Complex. The target complex specifies the type of target, the value of the target, and the target defenses.

a) Target Type: Targets are classified as either point, area, or collateral targets. The first two classes are the most commonly used. A target is considered a point target when a single well-placed weapon can kill it, an example being a missile silo. If more than one weapon is required to cover the target's area, such as a military base, it is considered an area target.

Collateral targets, however, are a collection of point

targets or an area target that has been broken up into individual aim-points. In contrast to a collection of point targets considered to be independent in that no single weapon can kill more than one target, collateral targets are targets located close enough together that all of them can be destroyed by a single weapon. Including collateral targets significantly increases the complexity of a model because the point targets that are collateral targets are not statistically independent.

Targets can also be classified according to target characteristics. Two commonly used classes of targets are Value and Force type targets. Force targets are those targets which if not destroyed have the ability to do immediate damage to the opponent. These include such targets as ICBMs and bomber bases. Value targets do not threaten the opponent but do have value to the attacker. These include targets such as factories and cities.

b) Target Value: The usual measure of effectiveness with which different weapon allocations are compared is the expected target value killed. The more complex the calculation of target value, the more complex the model. The simpler models either do not address target value or assume that all targets are given the same value of one unit. More complex models prioritize the targets as to their importance of destruction. The most complex models assign numerical values to the targets; however, there is a serious lack of

methodology in generating target values (Ref 27:13).

c) Target Defenses: The simplest models either assume no defenses or include the defenses in the form of penetration probability. More advanced models treat defenses as a specified attrition of attacking weapons before a target can be damaged. In many models the defenses are simulated by a specific probability of penetration for each target complex versus a particular type of weapon (Ref 27:14).

Model Terminology. The following terminology is used extensively in the literature.

a) Damage Expectancy (DE): DE is the statistical expected value that the weapon will arrive, detonate, and cause a certain level of damage (Ref 24:11).

b) Hedge: A hedge is an input rule or constraint that allows the analyst to specify auxiliary side goals or extra requirements while maintaining the ability to allocate in accordance with the main objective.

c) Force Optimization: Force optimization is concerned with the determination of the appropriate force structure to accomplish a set of specified goals subject to force availability, flexibility, and budget constraints.

d) Target Optimization: Target optimization consists of allocating weapons against targets in such a way as to obtain an optimal employment of the weapon mix. Normally this involves an allocation that attempts to maximize the damage achieved to a set of targets.

REVIEW OF EXISTING MODELS

The following list of analytical models and various mathematical formulations were examined to determine the essential elements of various models. This examination helped in the conceptualiztion and formulation of functions and formulas used in BRIK.

Arsenal Exchange Model. The Arsenal Exchange Model (AEM), developed by Martin-Mariettta Corporation, is the most widely used strategic analysis model. It was initially built in 1964 as a cost effectiveness model for use in advanced ICBM studies. But, in response to user requirements, it has evolved into a large expected value model used to study the structure of strategic forces. As of the last revision, documentation published in March, 1982, the AEM requires 1500k of computer storage.

AEM enjoys tremendous flexibility in its allocation rules. It permits analysis of different types and levels of exchanges and is constructed to accept a variety of strike objectives. The analyst also has the ability to structure a variety of employment constraints input as hedges. These hedges fall into three separate categories (Ref 3:10):

1) A specified amount of value destroyed on a particular target set must be accomplished by a particular weapon set.
2) A specified amount of weapons must be used against a particular target class.

3) A target set can not be attacked by more than a certain amount of a particular set of weapons.

The model employs a singleshot kill probability against point targets as

$$SSPK = 1 - .5^{(R_{ij}/CEP_i)}$$

where R_{ij} is the lethal radius of weapon i when it is allocated against target j, CEP_i is the circular error probable of weapon i. For area targets, AEM adjusts the point target SSPK as follows:

$$SSPK_{area} = SSPK\left[\frac{X}{C}(1-(1-\frac{X}{C})^B)+(1-\frac{X}{C})(\frac{X}{C})^{(1/B)}\right]$$

where X, B, and C are shape parameters resulting from a regression by RAND (Ref 28:33). Area and terminal defenses are also incorporated in the model.

The optimal allocation was attained by a goal-oriented Integer Linear Program using the Lagrangian method. However, because users wanted to trade off integer solution assurance for greater computational efficiency, the model has been converted to a multi-objective model solved by a linear programming (LP) method of allocation.

BSWADE. The Strategic Weapon Allocation Damage Expectation (SWADE) model was developed as an "in-house" project by Headquarters SAC/XPS in the mid-1970's and was designed to have a fast run time (Ref 12:35). Consequently, it aggregates and clusters much of the targeting data so that

data manipulation is kept to a minimum. In its current form, there are two separate programs, BSUDS and BALLOT.

The Single Uniform Data System (SUDS) is a user-friendly program that allows an analyst to enter data from a card deck (Ref 1). It takes all of the pre-specified force structures and allocation rules and converts them into a form which is capable of being used in the allocation phase. The model uses the VNTK target vulnerability system to determine its SSPKs.

The allocation phase uses a computer program called BALLOT, a version of an earlier program called ALLOT (Ref 23). It uses linear programming techniques to determine the optimal laydown of a nuclear arsenal against an aggregated target base, where optimal is defined as attaining the maximum DE of the entire target complex. It is a one-sided exchange where the optimal weapon-to-target assignment can be found according to one of the following objectives:

1) The total value destroyed is to be a maximum.
2) The total number of weapons employed is to be a minimum.
3) The total megatonage employed is to be a minimum.
4) The total EMT (Equivalent Megatonnage) employed is to be a minimum.
5) The total CMP (Countermilitary Potential) is to be a minimum.

BALLOT has the ability to accept manually-inserted constraining equations into the LP (Ref 23:8) and, as was

mentioned before, it has the ability to declare certain target classes as being time urgent.

Hodson. This paper describes an approach to weapon allocation. The authors point out that the general problem of weapon allocation is that "of assigning weapons to targets in such a way as to achieve a desired level of destruction". The authors go on to say that "when different combinations of weapon types and numbers are applied to the same target system with the same targeting philosophy, it is possible to measure the effect of adding, retaining, or deleting certain weapon systems" (Ref 17:2). The authors use linear programming to determine an optimal allocation. If S_{ij} represents the single shot probability of survival (SSPS) of a type j target assigned a type i weapon, the expected survival probability of that target would be

$$S_{ij}^{X_{ij}}$$

where X_{ij} is the number of type i weapons assigned to the type j target. Hodson linearizes this survival function so that LP can be used. By using the monotonic characteristics of the logarithmic function,

$$\log[\prod_{i}(S_{ij}^{X_{ij}})]$$

becomes $\Sigma[(\log S_{ij})X_{ij}]$.

This model assumes that SSPS's are known and weapon and target characteristics are not discussed. Also, since no

attempt is made to keep X_{ij} an integer, this allocation method is only an approximation. However, the authors present analysis that concludes the error from this approximation is small.

Wambsganss. In an Air Force Institute of Technology thesis, Michael C. Wambsganss developed a framework that demonstrated that a realistic and flexible model could be built and still be manageable in terms of limited computer resources. The model could address both weapon performance and cost issues.

This framework was developed by extracting features of various nuclear exchange models in existence. The formulae to compute the single-shot probabilities of survival (SSPS) for various weapon/target combinations was acquired from AEM. To incorporate fast run time, a linear programming technique was used. To linearize the damage function

$$S_{ij}^{X_{ij}}$$

where S_{ij} is the single-shot probability of survival for target j when attacked by weapon i, and X_{ij} is the amount of i weapons allocated to target j, Hodson's logarithmic function was used.

Wambsganss introduced the idea of linear goal programming to allow consideration of multiple objectives. The goal programming methods used were weighted goal programming for the weapon allocation and compromise programming when

investigating cost issues.

The objective function for the allocation problem is

$$\text{Minimize } \Sigma W_j M_j$$

where W_j is the weight or value associated with goal j, and M_j is the variable representing the underachievement of goal j (Ref 27:79). The model Wambsganss developed required considerable amount of manual input of data.

This model and the three models previously discussed were the primary sources of information and techniques for the authors. The following models, although not used as extensively in this research, are included for the reader's benefit.

GROTTE. This is a two-strike model in which Red employs all its weapons against both Blue's force and value targets, and Blue retaliates with its surviving arsenal against Red's value targets. The model can handle four weapon types for both Red and Blue. SSPKs for each side are expressed as functions of reliability, penetration probability, and single-shot probability of kill with the damage function assumed to have an exponential nature. The model optimizes the damage difference between Blue's and Red's strikes using Fauld's branch and bound algorithm as a piece-wise linear approximation to the damage difference (Ref 15).

CODE 50. The is a widely used aggregated effectiveness model developed by Lambda Corporation which can handle a mixture of weapon types, permits up to three strikes and is

restricted to 48 target types. The model considers area and terminal defenses and contains submodels to determine missile and bomber penetration probabilities as well as weapon/target allocation. Because of the complexity of the damage function and the nonlinearity of portions of the constraint set, a Lagrangian solution method is used (Ref 15).

DAY. This model splits the general allocation problem into two problems, small and large. The small problem is to allocate weapons within target complexes, while the large problem coordinates the allocations and assigns an optimal desired ground zero (DGZ) and height of burst (HOB) to each weapon in the weapon stockpile. The model is complex and requires the solution of a nonlinear programming problem for every target complex. DAY assigns values for each target in the complex (Ref 11).

SOLAND. This model examines discrete ABM levels so as to minimize any damage inflicted from an optimal attack by an opponent. Both point and area defenses are considered and are assumed to be in the exhaustive mode, which implies that no damage can be inflicted on a target until the defenses have been exhausted. The attacker is assumed to know the ABM levels and a min-max approach is used, which is to minimize the weapons necessary to exhaust the defenses and to inflict as much damage as possible. There are bounds which specify the number of missiles that can be allocated against any particular target (Ref 25).

MANNE. This paper assumes that all weapons are homogeneous. The model uses X_{ij} as the probability with which weapon i is allocated against target j and P_{ij} is the conditional probability of kill for weapon i against the target j. The model minimizes

$$\sum_j V_j \prod_i (1-P_{ij} X_{ij})$$

where V_j is the worth of target j. Manne formulates a linear approximation of the above equation and solves the allocation using a transportation algorithm (Ref 19).

ARMS. This paper considers both a cost optimization model and a single-sided allocation model. Damage functions are determined via analytical fatality curves taken from the Code 50 model. The cost optimization model contains the allocation model as a submodel and considers only annual costs and production costs. Development costs are incorporated implicitly with these costs by assuming linear build-up of weapons over a specified time period. Both models consider 100 variables and are solved by the sequential-unconstrained-minimization technique (SUMT) (Ref 27:19).

BRACKEN. This is a convex programming model which optimizes an SLBM attack on bomber bases. The model is split into two stages. The first stage allocates the submarines to a set of feasible launch areas and the second stage is the actual allocation of SLBMs to bomber bases to maximize damage. Not all missiles are allowed to launch simultaneously, and some bombers are allowed to scramble safely as a

function of the missile's flight time and range. A standard, nonlinear programming routine was employed to yield the global maximum (Ref 7).

MIERCORT. This is a two-sided nuclear exchange model which allocates homogeneous missiles against cities that are defended by area and point defenses. The model operates in the exhaustive mode, allocating missiles to the defenses prior to attacking the cities. The allocation is achieved through a nonlinear integer progamming algorithm using a branch and bound technique (Ref 21).

DAVID. This model allocates a nonhomogeneous weapon base against an undefended target complex. The PSSK formula is a function of target hardness in PSI and the yield and CEP of the weapon. The damage function is linearized and through a method of separable convex functions can be solved as a transportation problem. By using a numerical example, the authors show that round-off errors are small when the number of weapons is much greater than the amount of targets, and the round-off error is large when the weapons are roughly equivalent or less than the number of targets (Ref 9).

PERKINS. This model develops optimal defensive strategies for an anti-missile defense and can be used for optimal offensive allocation strategies. The author contends that the success of defending units is proportional to the ratio of defending units to attacking units. A 12-step procedure is developed to determine optimal offensive and defensive

strategies. The method is fundamentally based on marginal rates of effectiveness which are represented by partial derivatives of the damage function. Allocations are made until diminishing returns are realized (Ref 22).

MATHESON. This is a game-theoretic approach to missile allocation in terms of costs or resources. There are four cases considered:

Case 1. Defense has no available resources.

Case 2. Defense has less than enough resources to equalize offense system costs.

Case 3. Defense has exactly enough resources to equalize offense system costs.

Case 4. Defense has more than enough resources to equalize offense system costs.

Employing game-theory min-max principles enables the model to generate explicit expressions for both offensive and defensive acquisition strategies in terms of unit costs. Unit cost coefficients for two offensive and two defensive systems are considered in the allocation.

Summary

The models examined range from very detailed to aggregated, and from scenario specific to generalized. Most models contain the same essential components and differ only in the assumptions associated with these components. The most "realistic" models express parameters such as probability of kill as functions of target and weapon input data.

It is the opinion of the authors of this thesis that the most useful models are those that permit multiple choices of objectives such as AEM and BSWADE.

III Model Characteristics

This chapter presents four characteristics of BRIK. The first section lists both model restrictions and scenario limitations and the second section presents all of the input requirements. The third section explains how single shot probability of survival (SSPS) is computed and the final part discusses the damage function.

Limitations

R.D. Specht quotes E. S. Quade as saying, "All of the assumptions of a model must be made explicit...that his (the modeler's) errors will be more evident" (Ref 26:219). While assumptions may not necessarily reflect errors, they do represent the modeler's perception of the "real world." Consequently, it is important for any user to be aware of judgements and limitations built into any program. Therefore, BRIK's limitations are explicitly brought forth, not to expose "error", but to provide a further understanding of the basis for the model's construction. This section is divided into two parts. The first part lists the restrictions inherent in the model, and the second part describes limitations that must be imposed on any scenarios. While these limitations are a direct result of the model's restrictions, they are included in a separate subsection to better express the shortfalls a user might find unacceptable.

Model Restrictions. The development and potential application of BRIK are based on the following items:

1) Aggregation. BRIK is a fully aggregated nuclear exchange model. All of the weapons and targets used for any allocation must be represented by classes. Within each class, any subsystem must have identical characteristics. For example, all items in a target class called BRIDGE must have the same input parameters of size, vulnerability, and value.

2) No individual targeting strategy. The output consists of the number of weapons from each weapon class that are used to attack a particular target class. If more than one weapon class is allocated to a single target class, BRIK does not provide the individual breakdown concerning which weapons go to which targets.

3) Non-integer solutions. No attempt is made to keep the weapon numbers in whole units. For example, the model could suggest assigning 24.6 weapons of type i to target class j.

4) No footprinting restrictions. In reality, multiple independently targeted re-entry vehicles (MIRV) from a single missile are restricted in the degree of dispersion of their targets. BRIK does not account for this.

5) Prompt damage only. The only nuclear effects considered when computing SSPS are damage due to overpressure, dynamic pressure, blast duration, and cratering. Prompt and residual radiation from thermal effects, neutrons, and gamma rays is not considered.

6) No collateral damage. Each target in each class is assumed to be separate. For example, if two targets are close to each other, destruction of one will have no effect on the other.

7) No nuclear fratricide. There exists the possibility that nuclear explosions will adversely affect the detonation of subsequent warheads. This is called fratricide. BRIK does not take fratricide into account.

Scenario Limitations. Any research design concerning the study of nuclear forces must consider scenarios. A force structure that looks good under one plan may be totally ineffective in another. For example, a force composed entirely of very accurate land-based missiles may look extremely impressive if the scenario calls for those missiles to attack known targets. However, if the current policy dictates they will only be used for retaliation, the force would be very ineffective if they could be destroyed in a first strike. Since scenario is so important, the limitations imposed on any attack scheme are explicitly stated.

1) The target and weapon base must be completely known. The number, diameter, value, and some expression of vulnerability must be known for each target. For each weapon, the user must know the number of delivery vehicles, the number of warheads per weapon, circular error probable (CEP), the yield of each warhead, a reliability figure expressing the

probability that the weapon will successfully arrive at its destination and detonate, and daily and generated alert rates.

2) Any exchange can only be represented as a sequence of attacks. Before Blue can retaliate to a Red strike, it must be assumed that the Red strike has been completed.

3) Time is not explicitly modeled. The model does not account for the fact that some weapons will detonate earlier than others. However, it is possible to restrict some weapons from attacking certain targets. This permits consideration of time urgent targets.

4) Point defenses are not considered. The advantages gained from a defensive system placed only in one geographic location cannot be studied with BRIK. However, the reliability percentage of an individual weapon class can be used to study the effect a particular area defensive system has on the overall problem. For example, if it is determined that a particular defensive system degrades a weapon class performance by 20%, this percentage can be included in the reliability factor, REL, for that particular weapon. However, this factor will now apply whether the weapon's attack is in a heavily defended area or on an undefended target.

5) Imprecisely located targets cannot be studied. One of the assumptions of BRIK is the complete description of the target and weapon bases. While coordinates of each target are not an input requirement, it is assumed that the location of each target is known.

6) Command and Control is not considered. BRIK does not account for the effects from loss of any positive control elements.

This concludes the limitations the model places on the user. Next the input requirements are presented.

Input Requirements

To perform an allocation, BRIK requires complete description of the weapon and target base. Table I lists the input requirements for the weapon complex. Table II-A gives the input requirements for each target class and Table II-B lists the additional input requirements if BRIK is asked to compute a value associated with force targets. Using this data, BRIK internally computes an SSPS for each weapon/target combination. The methods used for the SSPS computations and the appropriate formulas are described in the next section.

Single Shot Probability of Survival

Most nuclear exchange models, including AEM, consider only prompt damage mechanisms to compute SSPS. Even though this underestimates the total destruction resulting from a nuclear detonation, the vogue of using only prompt damage will be continued with this effort.

This model has the capability to consider four prompt damage effects when computing SSPS. These effects are overpressure, dynamic pressure, blast wave duration, and cratering. Overpressure is the transient pressure, usually expressed in PSI, exceeding the ambient pressure, manifested in the

TABLE I
Input Requirements for the Weapon Complex

wpname(i)	Name of each weapon class i
numwpn(i)	Number of delivery vehicles in each class
numwrh(i)	Number of warheads per delivery vehicle
rel(i)	Probability weapon class i will reach its target and detonate
wpncep(i)	Circular error probable (CEP) of weapon i in nautical miles
wpnyld(i)	Yield in megatons of each weapon i
wpndda(i)	Daily alert rate of weapon class i
wpnga(i)	Generated alert rate of weapon class i

TABLE II-A
Input Requirements for the Target Complex

tgtnam(j)	Name of each target class j
numtgt(j)	Number of targets in each class j
tgtpsi(j)	Vulnerability of each target class j expressed as a VNTK or PSI number
tgtdia(j)	Diameter in nautical miles that encompasses 95% of a target in class j
tgtcat(j)	Target category. F, V, or M for force, value or military.
ntgtpr(j)	Preemptive priority of target class j
tgtval(j)	Number reflecting the value of each target in class j

TABLE II-B
Input Requirements to Compute Force Target Value

tgtrel(j)	Reliability of weapons at target class j
tgtcep(j)	CEP of weapons at target class j
tgtyld(j)	Yield per warhead in megatons of each weapon at target class j
ntgtwh(j)	Number of warheads located at target class j

shock wave from an explosion. Dynamic pressure is the air pressure which results from the mass air flow (or wind) behind the shock front of a blast wave. Blast wave duration is the time either the overpressure or dynamic pressure is present, and cratering is the pit, depression, or cavity formed in the surface of the earth by a surface or underground explosion (Ref 14:632-637). A description of the methods available to consider these damage effects and the formulas that appear in BRIK are presented next.

Vulnerability. The vulnerability of a target class can be expressed either as a VNTK figure or a PSI number. Depending on the input from the user, BRIK computes the SSPS of a weapon/target combination from one of two methods. The first method is called the Physical Vulnerability System. In this system, a target's susceptibility to blast damage is indicated by a combination of numbers and letters. The vulnerability number, VNTK, consists of a two-digit number reflecting the target hardness relative to a specified damage level, followed by a letter indicating predominant sensitivity to overpressure (P) or dynamic pressure (Q), then finally a K factor. The K factor allows for hardness adjustments to be made to account for the effects of variations in blast wave duration due to different weapon yields (Ref 5:34). The following example indicates the importance of considering blast wave duration. If the overpressure duration is short (1 to 3 milliseconds), overpressures between 390 PSI and 470

PSI are required to achieve a 50% mortality rate in humans. However, animal tests have shown that, if the overpressure duration is long (80 to 100 milliseconds), 40 to 50 PSI will produce that same 50% mortality rate (Ref 8:7-92). Because buildings are also susceptible to blast duration, the VNTK system has been included in this model. However, VNTK numbers may not always be available; therefore, the less accurate method of designating an estimated PSI level has also been included. This entails estimating the peak overpressure in pounds per square inch (PSI) that will provide the desired level of damage. A discussion concerning appropriate PSI numbers can be found in Reference 27, page 31.

VNTK. The ALLOT computer program, currently a part of the SAC BLUE SWADE model, uses the VNTK system to compute SSPS. A listing of BLUE SWADE was acquired and DPDX, the subroutine that computes probability of kill (PK), was extracted. After a few minor modifications, that subroutine was placed in BRIK as Function VTK. It is known that LtCol Donald J. Berg wrote Allot (Ref 23:100); however, little is known about the regression techniques used to develop the formulas in DPDX. It is certain, though, that the subroutine is being used properly (see Verification in Chap VI). First, this section will discuss the input requirements for DPDX and how it is being used in BRIK. Then, the formulas and the data table in the subroutine will be listed.

The parameter statement associated with DFDX in ALLOT contains ten elements. The first three--VN, T, and K1--are the three parts to the VNTK number input by the user. The next three elements concern weapon and target characteristics. They are 95% of the radius of a circle that encompasses the target, weapon yield, and weapon CEP. The seventh element, H, is always either zero or one. It denotes whether the weapon detonation was an air burst or a ground burst. The last three elements concern numbers computed by the subroutine. The first one is probability of kill and the last two are used in a sensitivity printout in the output section of the program. In revising the subroutine, the two elements associated with the sensitivity output were removed. Also, the equations that computed those figures were taken out. Next, a loop was included inside the subroutine to compute PK assuming the attacking weapon detonated as both an airburst and a ground burst. This negated the requirement to designate whether or not a weapon detonated in the air or on the ground. To compute SSPS, BRIK uses the highest PK computed from these two cases. Following are the formulas and the data tables used in Function VTK.

PK = EXP(-EXP(b(1,T)+U*(b(2,T)+U*(b(3,T)+U*b(4,T)))))

where

T = 1 if target is susceptible to over-pressure

T = 2 if target is susceptible to dynamic pressure

U = S/(XD*C)

XD = $1/Y^{(1/3)}$

C = $\sqrt{CEP^2 + .231*R^2}$

R = radius of a circle in feet that encompasses 95% of the target

S = EXP(a(1,H,T)+V*(a(2,H,T)+V*(a(3,H,T)+V*(a(4,H,T)
+V(a(5,H,T)+V*(a(6,H,T)+V*a(7,H,T)))))))

H = 0 or 1 representing airburst or ground burst

V = VN+d(1,K,T)+XD*(d(2,K,T)+XD*(d(3,K,T)+XD*d(4,K,T)))

K = K Factor

Y = Yield in Kilotons

CEP = Circular Error Probable.

It should be noted that there are limits in the subroutine. If U is greater than 10, PK is automatically set to 1. Also, if U if less than 0.1, PK is assigned 0.

The final step in acquiring SSPS is to include weapon reliability. The following formula is used for each weapon target combination:

$$SSPS_{ij} = 1 - REL_i * PK$$

where $SSPS_{ij}$ represents the single shot probability of survival of a target in class j assigned a weapon from class i, and REL_i is the reliability of weapon class i.

There are three arrays in the above formulas that obtain their data from tables. Array a(7,2,2) is a three-dimensional array that can be thought of as having two sets of two rows of numbers, each row having seven elements. Those 28 numbers are in Table III. Array b(4,2) has two rows of four

numbers each. Its elements are in Table IV. Finally, array d(4,10,2) has two sets of ten rows, each row having four numbers. These numbers are listed in Table V.

This concludes the description of Function VTK. BRIK automatically distinguishes between a VNTK number and PSI. If a PSI number is input, Function PK is used to compute SSPS. The formulas in that function are presented next.

PSI. To compute SSPS given a PSI number, the model starts by computing a lethal radius (LR). If PSI is greater than 10, the lethal radius resulting from a weapon of class i attacking a target of class j is expressed as follows (Ref 27:49):

$$LR_{ij} = 2.8Y_i^{(1/3)}(PSI_j-7.37)^{-.352} \quad (1a)$$

If PSI is less than or equal to 10 the lethal radius is (Ref 10:214)

$$LR_{ij} = (6.81*Y_i^{(2/3)})/PSI_j^{.62} \quad (1b)$$

where Y_i equals yield in megatons. For cratering, it is assumed that the target is destroyed if it is inside the lip crest of a crater formed in soft rock. The formula to compute this LR is derived in the following manner. The apparent crater radius for a one-kiloton surface burst in dry soft rock is 61 feet. Scaling the yield to the .3 power and computing the lip crest radius,

$$R_{ij} = 1.25*61*Y_i^{.3}$$

where Y_i equals yield in kilotons and R_{ij} equals the crater lip crest radius in feet (Ref 14:253). In BRIK, the formula

TABLE III

Data for Array a(7,2,2)

```
a(x,x,1)
8.214,-.1118, 5.265e-4, 2.162e-5,-6.638e-7, 7.132e-9,-3.064e-11
8.783,-.1355, 2.355e-3,-2.086e-4, 9.901e-6,-1.872e-7, 1.227e-09
a(x,x,2)
8.315,-.1033,-7.908e-4,-9.039e-5, 1.458e-5,-5.220e-7, 5.726e-09
8.789,-.1120,-6.658e-5,-5.803e-4, 5.853e-5,-1.905e-6, 2.056e-08
```

TABLE IV

Data for Array b(4,2)

```
   1.66904, -2.17442,  .260926, -.0752178
   1.65946, -2.15466,  .295853, -.0484718
```

TABLE V

Data for Array d(4,10,2)

```
  d(x,x,1)
  0.00000,  0.00000,  0.00000,   0.00000
- 0.577874, 1.56916,  0.0013762, 0.0063101
- 1.22391,  3.32978, -0.0020688,-0.0469539
- 1.957,    5.35163, -0.0477323,-0.154546
- 2.80456,  7.74241, -0.225298, -0.345665
- 3.81168, 10.7222,  -0.84178,  -0.513504
- 5.05081, 14.6336,  -2.37475,  -0.476885
- 6.65796, 20.1955,  -5.98136,   0.315062
- 8.92061, 28.9801, -14.2174,    3.11471
-12.7265,  45.9954, -35.5644,   12.2164
  d(x,x,2)
  0.00000,  0.00000,  0.00000,   0.00000
- 0.288917, 0.79886, -0.0399065, 0.0011525
- 0.611921, 1.72876, -0.18691,   0.0095419
- 0.978187, 2.83679, -0.511558,  0.0514480
- 1.40112,  4.19642, -1.13330,   0.18118
- 1.90063,  5.91548, -2.23005,   0.484749
- 2.50907,  8.17913, -4.11606,   1.12029
- 3.28374, 11.3172,  -7.35613,   2.37622
- 4.34296, 16.0486, -13.2112,    4.90666
- 6.01000, 24.4300, -25.3672,   10.643
```

used expresses radius in nautical miles and yield in megatons. Therefore,

$$LR_{ij} = \frac{1.25*61}{6080}*(Y_i*1000)^{.3}$$

or,

$$LR_{ij} = .125*Y^{.3} \qquad (2)$$

Cratering is independent of PSI. Therefore, this formula places a ceiling on the protection gained from super-hardening. Figure 1 plots the relationship between weapon yield and the PSI ceiling. For PSI values below this ceiling, BRIK uses formula set 1 to compute LR. For PSI values above this ceiling, the lethal radius is computed by formula 2.

Using one of the above lethal radii formula, the circular normal distribution is used to compute the single shot probability of kill (SSPK) for a point target in class j allocated a weapon from class i. The formula is as follows:

$$SSPK_{ij} = 1-.5^{(LR_{ij}/CEP_i)^2}$$

where CEP_i equals the circular error probable of weapon i and LR_{ij} equals the larger of the two lethal radii computed above.

This formulation assumes what is commonly called a "cookie-cutter" damage function: if the warhead lands at or within the lethal radius, the target is completely destroyed; if the warhead lands outside the lethal radius, the target is completely undamaged. However, in reality, this does not happen. Because of variability in target hardness, warhead effects, and the hardness of different target

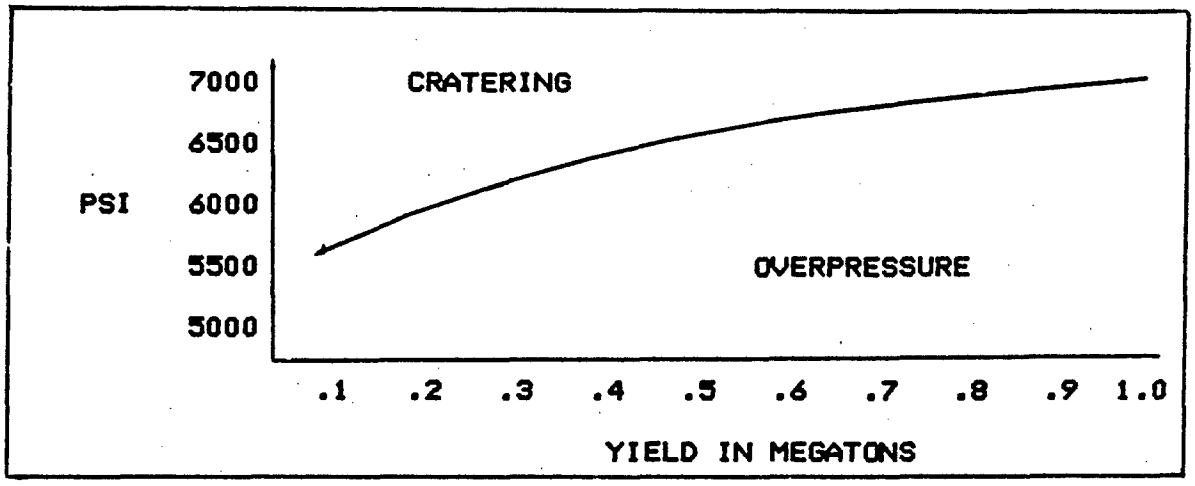


Figure 1. Boundary Between LR Formulas of Cratering and PSI

components, there is some region around a target where a weapon can be detonated and the target is neither completely destroyed nor completely unharmed. To represent this region, a model could use a log-normal damage function. A technique, sometimes referred to as a quadrature scheme, is used to compute SSPK. Given CEP and LR, points are established at known distances from a target. The probability that the weapon hits each point is multiplied by the probability that the weapon destroys the target, assuming the weapon hit each point. The average of these products provides the probability of kill. This technique is not used here because the added complexity of computing SSPK using a log-normal function results in an insignificant change in probability of survival--about 2.2 percent (Ref 4:16).

If the area of the target class j is large in comparison to a weapon's lethal diameter, a different interpretation of damage is needed. For a point target in class j, $SSPK_{ij}$ is

the probability a weapon i will detonate with the target located within the weapon's lethal radius. However, for an area target j, $SSPK_{ij}$ can be interpreted as the percentage of the target destroyed if allocated one warhead of type i. To account for this change of interpretation, an adjustment is made to the $SSPK_{ij}$ computed for a point target. The following formula was found in Reference 27, page 41, credited to Reference 20:

$$P(x) = SSPK_{ij}\left[\frac{X}{C}\left(1-\left(1-\frac{X}{C}\right)^{B}\right)+\left(1-\frac{X}{C}\right)\left(\frac{X}{C}\right)^{(1/B)}\right]$$

where

$X = (L/D)$

L = Lethal Diameter

D = Target Diameter

$$C = \begin{cases} C_1 C_2^{\,y_y C_3} & 0<y<1 \\ 1 + C_4 C_5^{\,y_y C_6} & 1<y<\infty \end{cases}$$

$$B = \begin{cases} C_7 C_8^{\,y_y C_9} & 0<y<2 \\ C_{10} - C_{11} C_{12}^{\,y_y C_{13}} & 2<y<\infty \end{cases}$$

$y = LR/CEP.$

The values of c1 to c13 are as follows:

c1 = 7.45995	c5 = .0930395	c9 = -.29298	
c2 = .207535	c6 = 3.83323	c10 = 2.73109	
c3 = 2.4262	c7 = 1.689	c11 = 1.12109	
c4 = 5.89243	c8 = 1.09218	c12 = .79865	
		c13 = .639752	

The final step to acquire SSPS includes weapon system reliability. Therefore,

$$SSPS_{ij} = 1\text{-}REL_i * SSPK_{ij}$$

where $SSPS_{ij}$ is the single shot probability of survival of a target in class j assigned a weapon from class i, and REL_i is the reliability of weapon class i.

This concludes the presentation of BRIK's computational techniques for probability of survival. How the single shot probability of survival for a target in class j assigned a weapon in class i is used in the damage function is discussed next.

Damage Function

The damage function in BRIK works with expected value concepts. Specifically, the measure of merit involves the expected percentage of targets in a class that survives an allocation. For example, let S_{ij} denote the survival probability of a class j target if allocated a single type i weapon. If more than one type i weapon was assigned to target j, the survival probability would become

$$S_{ij}^{X_{ij}}$$

where X_{ij} represents the number of class i weapons assigned to target j. If there were several different types of weapons that could be assigned to this single target, the survival probability could be expressed as

$$\prod_i S_{ij}^{X_{ij}}$$

where S_{ij} represents the survival probability of target j if

allocated a type i weapon. For example, if target j has an SSPS of .4 if assigned a type 1 weapon and .3 if assigned a type 2 weapon, the probability that target j would survive if assigned one weapon of each type is

$$\prod_{i=1}^{2} S_{ij}^{X_{ij}} = (.4)^1(.3)^1 = .12$$

Next, assume that there are two more targets in class j for a total of three targets. Also, assume that the second target was allocated one type 1 weapon and that the third target was allocated one type 2 weapon. The probability of survival for the second target is

$$\prod_{i=1}^{2} S_{ij}^{X_{ij}} = (.4)^1(.3)^0 = .4$$

and the probability of survival of the third target is

$$\prod_{i=1}^{2} S_{ij}^{X_{ij}} = (.4)^0(.3)^1 = .3.$$

The average probability of survival of the entire class is

$$(1/3)(.12+.4+.3) = .273.$$

Expressed mathematically, the survival function for class j is

$$(1/a_j) \sum_{k=1}^{a_j} \left(\prod_{i=1}^{m} S_{ij}^{X_{ik}} \right)$$

where a_j represents the number of targets in class j and m equals the number of weapon classes. This expression for the average probability of survival is the sum of products and is difficult to deal with in terms of solvability. Therefore, the following approximation is used in BRIK. The average probability of survival of the entire class j can be aproximated by

$$\prod_{i} S_{ij}^{X_{ij}}$$

where X_{ij} represents the average number of weapons allocated against each target in class j. In the current example, a total of two weapons from each weapon class was assigned to the three targets in class j. Using the approximation to compute the average probability of survival yields

$$\prod_{i=1}^{2} S_{ij}^{X_{ij}} = (.4)^{(2/3)}(.3)^{(2/3)} = .243$$

This compares with .273 obtained from using formula (1). If X_{ij} for all i are integers, there is no error, and the largest error found occurs when X_{ij} is equal to .5, 1.5, 2.5, etc. Both Reference 27 and Reference 17 agree that the magnitude of the error in this approximation is "believed to be small." This belief combined with the increase in solvability convinced the thesis authors to use it in BRIK.

In conclusion, the damage function employed in BRIK is an approximation that expresses the probability of survival of a target class when allocated different weapons. How that function is used in the linear constraints of a goal programming formulation is discussed in Chapter IV.

IV Constraint Development

BRIK uses a linear goal programming routine to allocate weapons to targets. Goal programming requires two things, an objective function and a constraint set. A discussion of goal programming and a presentation of the objective functions in BRIK will be found in Chapter V, while this chapter derives all of the constraints used by BRIK in the linear programming matrix. Regardless of the problem, the constraint set is identical. This set consists of one constraint for each target class, one constraint for each weapon class, one constraint for the extreme goal, and one constraint for each hedging option designated by the user. As currently dimensioned, BRIK can handle a maximum of 61 constraints.

Target Constraint

Let S_{ij} represent the single shot probability of survival of a target in class j allocated one weapon of type i. If X_{ij} represents the number of weapons i allocated to each target in class j, the survival of target j can be represented by

$$S_{ij}^{X_{ij}}.$$

If different types of weapons i are allocated to target class j, survival of each target becomes

$$\prod_i S_{ij}^{X_{ij}}.$$

In a goal programming problem, a constraint containing the survival expression and a damage expectancy (DE) goal can be written as follows:

$$\prod_i S_{ij}^{X_{ij}+n_j-p_j} = 1-DE_j = PV_j$$

where PV_j equals $(1-DE_j)$, the survival goal of a target in class j; n_j equals the negative deviation from the survival goal; and p_j equals the positive deviation from the survival goal. If this constraint was used in a non-linear goal programming problem, minimizing p_j would drive the survival expression to the goal PV_j.

It is possible to linearize this constraint to permit its use in a standard linear programming model. By moving the goal variables n_j and p_j to the right-hand side,

$$\prod_i S_{ij}^{X_{ij}} = PV_j - n_j + p_j.$$

If PV_j is divided out of each term in the right-hand side,

$$\prod_i S_{ij}^{X_{ij}} = PV_j\left(1+\frac{-n_j+p_j}{PV_j}\right).$$

Now, if the natural logarithm of each side is taken, the following is attained:

$$\ln\left(\prod_i S_{ij}^{X_{ij}}\right) = \ln\left[PV_j\left(1+\frac{-n_j+p_j}{PV_j}\right)\right].$$

Because the logarithm of products is the sum of logarithms,

$$\sum_i \ln(S_{ij}^{X_{ij}}) = \ln PV_j + \ln\left(1+\frac{-n_j+p_j}{PV_j}\right)$$

and

$$\sum_i \ln(S_{ij}^{X_{ij}}) - \ln\left(1+\frac{-n_j+p_j}{PV_j}\right) = \ln PV_j.$$

The survival function will always be greater than zero and less than or equal to one. This is also true of a goal PV_j. Since the logarithm of any number between zero and one is negative, it is necessary to multiply this constraint by negative one to obtain a positive right-hand side:

$$-\sum_i \ln(S_{ij}^{X_{ij}}) + \ln\left(1+\frac{-n_j+p_j}{PV_j}\right) = -\ln PV_j.$$

It is also possible to bring the exponent X_{ij} outside the

parenthesis and obtain

$$-\sum_i(\ln S_{ij})X_{ij}+\ln\left(1+\frac{-n_j+p_j}{PV_j}\right) = -\ln PV_j.$$

In a goal programming problem, n_j represents the negative deviation from a goal and p_j represents the positive deviation from a goal. If a goal is not met exactly, it is impossible to simultaneously deviate from that goal in both a positive and negative direction. Therefore only one of the variables, n_j or p_j, can be non-zero at any given time. Thus, either n_j or p_j must always be zero. Since $\ln(1) = 0$, it is possible to split the second term in the above equation and obtain

$$-\sum_i(\ln S_{ij})X_{ij}+\ln\left(1+\frac{p_j}{PV_j}\right)+\ln\left(1+\frac{-n_j}{PV_j}\right) = -\ln PV_j$$

or

$$-\sum_i(\ln S_{ij})X_{ij}+\ln\left(1+\frac{p_j}{PV_j}\right)-\left(-\ln\left(1+\frac{-n_j}{PV_j}\right)\right) = -\ln PV_j.$$

This model is an aggregated nuclear exchange model. Therefore, let S_{ij} represent the survival probability of all class j targets allocated class i weapons. Since X_{ij} equals the average number of weapons i allocated to each target in class j, it is possible to perform a variable transformation of X_{ij}, such that

$$X_{ij} = Y_{ij}/N_j$$

where Y_{ij} represents the number of class i weapons allocated to target class j and N_j equals the number of targets in class j. Using this transformation greatly reduces round-off error resulting from non-integer programming; however, it introduces the approximation errors mentioned in

Chapter III. This transformation yields the following:

$$-\sum_i(\ln S_{ij})(Y_{ij}/N_j)+\ln(1+\frac{p_j}{PV_j})-(-\ln(1+\frac{-n_j}{PV_j})) = -\ln PV_j$$

or

$$-\sum_i(\ln S_{ij})Y_{ij}+N_j\ln(1+\frac{p_j}{PV_j})-(-N_j\ln(1+\frac{-n_j}{PV_j})) = -N_j\ln PV_j$$

Since $0 <= n_j <= PV_j$,

$$N_j\ln(1+\frac{-n_j}{PV_j})$$

is always non-positive. And, since $0 <= p_j <= 1 - PV_j$,

$$N_j\ln(1+\frac{p_j}{PV_j})$$

is always non-negative. Therefore, these two expressions can represent the deviational variables in the final form of the target constraint. The final form of the target constraint used in this effort is

$$-\sum_i(\ln S_{ij})Y_{ij}+d_j^- - d_j^+ = -N_j\ln PV_j$$

where d_j^- equals

$$N_j\ln(1+\frac{p_j}{PV_j})$$

and d_j^+ equals

$$-N_j\ln(1+\frac{-n_j}{PV_j}).$$

As mentioned above, minimizing p_j would drive the survival expression to the goal PV_j. Since p_j appears in the term that represents the negative deviational variable, d_j^-, minimizing d_j^- would drive this constraint to the right-hand side goal. Likewise, minimizing d_j^+ would effectively place an upper bound on the constraint, preventing any over-achievement of the designated goal. How these deviational variables are used in the different objective functions in BRIK will be discussed in the next chapter.

This is not the first time a non-linear damage function has been linearized with logarithms. Both Hodson and Wambsganss did so. Hodson's formulation had only a single deviational variable, sometimes referred to as minimizing the maximum deviation (Ref 16:178), and his goals were ratios of DE. For example, if there were three target classes and it was desired to have an allocation that would give DE in the ratio of 3:5:4, his PVj's were set as follows:

$$PV1 = \ln 3 \qquad PV2 = \ln 5 \qquad PV3 = \ln 4.$$

This worked satisfactorily for him (Ref 17:14). However, just as the constraints are being used here, Wambsganss used them to express DE goals in terms of a percentage of survival for each class j and each constraint had a different deviational variable (Ref 27:71-74). This sometimes produces some undesirable side effects. The cause of those side effects and possible solutions will be discussed in Chapter VII.

Weapon Constraints

In the target constraints, the decision variable Y_{ij} represents the number of type i weapons assigned to all targets in class j. The weapon constraints sum all of the weapons in class i assigned to each target class j. This sum is set equal to the number of warheads available in each weapon class i. This is depicted by the following:

$$\sum_j Y_{ij} = A_i$$

where A_i equals the number of warheads available in class i. To maintain the goal programming formulation, deviational

variables, d_i^- and d_i^+, are added. This gives the final form of the weapon constraints:

$$\sum_j Y_{ij} + d_i^- - d_i^+ = A_i.$$

The presence of these deviational variables provides a tremendous amount of flexibility. Both variables are used at various times in BRIK's objective functions. Minimizing d_i^+ prevents the use of more weapons than are available, while minimizing d_i^- forces the use of as many weapons as possible. As seen in the next section on hedging constraints, completely different interpretations of the same constraint result from switching the deviational variables that appear in the objective function.

Hedging Constraints

Hedging options permit a user to designate a variety of allocation rules. In its current form, the model has seven options to choose from. As currently dimensioned, up to 20 hedging constraints can be added. In this section, the seven hedging options and the form of their constraints are listed.

The following is a list of the seven types of hedging options available:

1) Enforce a minimum level of damage on a particular target class.

2) Enforce a minimum level of damage on a particular target class using a specific set of weapons.

3) Enforce an upper level of damage on a particular target class resulting from a specific set of weapons.

4) Restrict the number of a class of weapons which can be allocated to a target class.

5) Build your own constraint.

6) Enforce a minimum level of damage on a particular set of target classes.

7) Restrict the number of weapons which can be allocated to each target in a particular target class.

There is a constraint associated with each of the above list of options. For option one, the constraint is identical to a targeting constraint. Usually the DE entered here would be lower than the normal achievement goal.

$$\text{Min } \{d^-\}$$

st (subject to)

$$-\sum_i (\ln S_{ij}) Y_{ij} + d^- - d^+ = -N_j \ln PV_j$$

where the variables are as previously defined. For hedging type two, the constraint differs from type 1 in that not all weapon types must be in the constraint. The user can designate a subset k of weapons to be included in this hedge.

$$\text{Min } \{d^-\}$$

st

$$-\sum_{i \in k} (\ln S_{ij}) Y_{ij} + d^- - d^+ = -N_j \ln PV_j .$$

Type 3 hedge can be used to restrict the damage resulting from a designated set of weapons. There is no difference between this constraint and that of hedge 2. The difference comes from minimizing the positive deviational variable instead of the negative deviational variable.

$$\text{Min } \{d^+\}$$

st

$$-\sum_{i \epsilon k} (\ln S_{ij}) Y_{ij} + d^- - d^+ = -N_j \ln PV_j.$$

With any particular target class, a DE goal for the targeting constraint would normally be higher than the DE goal for this type 3 hedge. The purpose of this constraint is to further restrict the damage from some subset of weapons. It would not be needed if the DE goal on this constraint was set higher than the overall goal for the class.

Type 4 hedge sets the decision variable Y_{ij} equal to the maximum number of type i weapons that can be assigned to target class j. Deviational variables are included to preserve the G.P. formulation.

$$\text{Min } \{d^+\}$$

st

$$Y_{ij} + d^- - d^+ = \text{RHS}.$$

Type 5 hedge consists of customizing a constraint. For example, a user may have an unlimited number of two different types of warheads but a limited number of delivery systems. A constraint could be built that restricts the total number of the two warheads delivered. For example, assume there are already two weapon constraints in the problem such that

$$\sum_j Y_{1j} + d^- - d^+ = A_1$$

and

$$\sum_j Y_{2j} + d^- - d^+ = A_2$$

where A_1 and A_2 represent large warhead availabilities, but

the delivery vehicles for type one and type two warheads restrict the total number of deliverable warheads to A_3. The constraint would be as follows:

$$\min \{d^+\}$$

st

$$\sum_j Y_{1j} + \sum_j Y_{2j} + d^- - d^+ = A_3.$$

Minimizing d^+ would prevent A_3 from being exceeded. BRIK gives the user the ability to build any constraint. All that is needed is i, j, the coefficient of each required Y_{ij} term, and the appropriate right-hand side.

Type 6 hedge permits designating a minimum level of damage across a set K of targets.

$$\text{Min} \{d^-\}$$

st

$$-\sum_{j \in k} \left(\sum_i (\ln S_{ij}) Y_{ij}\right) + d^- - d^+ = -\left(\sum_j N_j\right) \ln PV_j.$$

Type 7 hedge restricts the number of weapons which can be allocated to each target in a particular target class. The constraint sums the variables that represent the weapons that can be assigned to some class j and sets that sum equal to some multiple M of the number of targets in class j.

$$\text{Min} \{d^+\}$$

st

$$\sum_i Y_{ij} + d^- - d^+ = MN_j.$$

This concludes the presentation of all seven hedging options included in the model. A great variety of allocation rules are thus permitted allowing tremendous flexibility.

Extreme Goal

A lowest priority goal is built into the constraint set to give the model the ability to drive the allocation to a unique solution. If there are weapons remaining following an allocation, there are an infinite number of allocations available that meet the attained DE goals. To insure that the allocation is unique, a lowest priority goal, an extreme goal, is included. The constraint can take one of two different forms,

$$\min \{d_e^+\}$$

st

$$\sum_i (\sum_j coef_i Y_{ij}) + d_e^- - d_e^+ = 0 \qquad (1)$$

where $coef_i$ is internally computed based on a choice of the user, or

$$\min \{d_e^-\}$$

st

$$\sum_i (\sum_{j \in k} Y_{ij}) + d_e^- - d_e^+ = \sum_i A_i \qquad (2)$$

where k is the set of targets chosen to receive the remaining weapons.

The user has five choices of extreme goals. They are

1) Minimize warheads used.

2) Minimize the amount of megatonnage used.

3) Minimize the countermilitary potential of the force.

4) Minimize the equivalent megatonnage of the force.

5) Use as much of the remaining arsenal as possible.

Constraint one is used for the first four choices and constraint two is used for choice five. If minimizing

warheads is chosen, $coef_i$ equals 1 in all cases. If minimizing megatonnage is chosen, $coef_i$ is equal to the megatonnage of weapon class i. If the user wishes to minimize the counter-military potential of the allocated force, $coef_i$ is a function of yield. If $y_i < 0.2$ megatons then

$$coef_i = y_i^{.8}/(6*CEP_i)^2$$

otherwise

$$coef_i = y_i^{.6}/(6*CEP_i)^2$$

(Ref 23:2). Finally, if it is chosen to minimize the equivalent megatonnage, the following is used. If $y_i < 1$ megaton then

$$coef_i = y_i^{(2/3)}$$

otherwise

$$coef_i = y_i^{(.5)}$$

(Ref 23:2).

To use as many of the remaining warheads as possible, the user must select the set K of targets. Currently, that set K of targets must be either the force targets, other military targets, value targets, or any remaining targets where the DE goals have not been met. After selecting the appropriate target set, constraint two is used for the extreme goal.

This concludes the derivation of all the constraints that can appear in a problem solved by BRIK. Some set of these constraints combined with an objective function constitute a linear goal programming problem. Goal programming, the objective functions, the constraint sets, and the solution algorithm will be presented in the next chapter.

V Model

BRIK is an interactive, goal programming model for nuclear exchange problems. The first section in this chapter reviews goal programming, providing an introduction to goal programming for those readers unfamiliar with the topic. It also offers a brief review for those readers fluent in goal programming techniques. Following the goal programming description are the mathematical formulations of BRIK. This section includes the various objective functions and the constraint sets that form the goal programming matrix. This chapter concludes with a description of PAGP, the preemptive goal programming algorithm used in BRIK. As mentioned in the preface, PAGP was acquired from the Association for Computing Machinery (ACM). It is copyrighted, therefore its commercial use is prohibited.

Goal Programming

Use of linear programming (LP) is restricted to a single overriding objective, such as maximizing total profit or minimizing total cost. However, this is not always realistic. Sometimes it is desirable to conduct studies that focus on a variety of other objectives, e.g., to maintain stable profits, increase one's share of the market, diversify products, or improve worker morale. An example of these statements that concerns many nuclear exchange models is the case where it is desirable to attain damage on a certain group of targets but also to minimize the weapons used in the

allocation. With a standard LP, there are two ways to formulate this problem. One, constraints could be made reflecting the desired damage expectancy (DE) of the respective target classes with the objective of minimizing the amount of weapons used. Or, two, the objective could maximize DE, but, to limit the amount of weapons used, constraints could be built which represent the amount of weapons available. In the first case, there is the possibility of having an infeasible problem. In the second case, an answer will be obtained, but it may not be the best answer because of the changing of the weapon constraints to limit warhead usage. This obstacle of multiple objectives that are incommensurate and often incompatible can be solved by providing a method of striving toward several such objectives simultaneously. That method is called goal programming (G.P.).

The basic idea is to establish a specific numerical goal for each of the objectives, formulate an objective function for each objective and then seek a solution that minimizes the weighted sum of deviations of these objective functions from their respective goals (Ref 16:172). Mathematically, the goal programming model reduces to the following:

$$\text{Min } Z = \sum_i (P_k W_{ik}{}^{+}d_i^{+} + P_s W_{is}{}^{-}d_i^{-})$$

st:

$$\sum_{j=1}^{n} m_{ij} X_j + d_i^{-} - d_i^{+} = g_i \qquad i = 1,\ldots,p$$

plus including any other linear programming constraints on the X_j. The variable g_i represents each objective's goal and

W_{ik}^{+} and W_{is}^{-} provide the relative weights of the deviational variables in the kth or sth ranking. The use of weights permits ranking of the variables at each priority assuming each variable is in terms of a single uniform measure.

P_k and P_s represent preemptive-priority factors. Any goal at preemptive priority k (designated by P_k) will always be preferred to (i.e., preempt) any goal at a lower priority k+1,...,K, regardless of any scalar multiplier W_{ik}^{+} or W_{is}^{-} associated with these lower priorities. For example, consider the decision procedure in purchasing a home. The buyer's first priority may be to consider only a home that is within a 20-mile radius from his or her place of work. All other homes are excluded from consideration. Next, the buyer may desire to limit the purchase price to under $100,000. Thus, even though there may be homes under $100,000 21 miles from work, they are excluded (or preempted) from consideration by the first priority. Thus, the preemptive-priority concept is used as an iterative screening process (Ref 18:380).

The symbols d_i^{-} and d_i^{+} represent deviational variables. The difference between what is accomplished and what is aspired to is the deviation from a goal. Having both a negative and a positive variable in the formulation permits both under- as well as overachievement of a goal. Sometimes it is useful to include these variables, occasionally called auxiliary variables, in standard simplex problems just to simplify the model. Auxiliary variables are simply variables

that are added to a model for convenience in addition to the original decision variables of the problem. "In most cases, this requires constructing an equation for each auxiliary variable that defines this variable in terms of the other variables in the model. This definition then is incorporated into the model by adding this equation as another equality constraint in the model. Including these equations"...allows incorporation of "auxiliary variables into the objective function in a linear programming format" (Ref 16:176-177).

In conclusion, the strength of goal programming comes from its flexibility. Many different aspects of a constraint set can be examined by simply changing the priorities of the objective terms. This will become apparent in the following section where only the objective function is changed to meet different problem requirements.

Mathematical Formulation

The feasibility of a model that had a linearized damage function with auxiliary variables in the target constraints had already been demonstrated in an earlier effort by Wambsganss (Ref 27). However, to give his model flexibility, his thesis presented many different formulations. To perform an allocation to meet damage goals, auxiliary variables from the target constraints appeared in a single objective function. To complete the model, hard constraints were added to represent weapon availability and any hedging options. Wambsganss cautioned that too many hedges may create a prob-

lem with no feasible solutions (Ref 27:77). In another formulation, to compute a possible reserve force, hard constraints were made out of both the target and weapon constraints. The terms in the objective function consisted of all of the decision variables. To preclude an allocation consisting of entirely one weapon, or the greater portion of a weapon class, Wambsganss suggested reducing the weapon availability prior to performing the allocation (Ref 27:80). Again, the possibility of an infeasible problem exists. These problems of infeasibility plus the programming complexity resulting from occasionally adding auxiliary variables motivated a search for a different formulation. The formulation eventually agreed upon used goal programming and appears in Figure 2.

Three advantages of goal programming are immediately apparent. One, the form of the constraint set does not change. Regardless of the problem being analyzed, every constraint contains deviational variables. The advantages gained from this standardization were realized during the flowcharting of BRIK. Knowledge of the objective function is not needed to keep track of which columns hold which decision variables. The second advantage of goal programming came from the objective function. All of the different allocations can be performed with just the deviational variables. It is now possible to minimize the weapons used in a problem without having the decision variables appearing in the

Objective Functions

1) $\min\ ((\sum_i d_i^+),(\text{Hedging}),(\sum_j W_j d_j^-),(d_e^+))$

2) $\min\ ((\sum_j d_j^+),(\text{Hedging}),(\sum_j d_j^-),(\sum_i d_i^-),(d_e^+))$

3) $\min\ ((\sum_i d_i^+),(\sum_j d_j^+),(\text{Hedging}),(d_e))$

Constraint Set

st:

$$-\sum_i (\ln S_{ij}) Y_{ij} + d_j^- - d_j^+ = -N_j \ln PV_j \qquad \forall\ j$$

$$\sum_j Y_{ij} + d_i^- - d_i^+ = A_i \qquad \forall\ i$$

or $\begin{cases} \sum_i (\sum_j \text{coef}_i\ Y_{ij}) + d_e^- - d_e^+ = 0 \\ \sum_i (\sum_{j \in k} Y_{ij}) + d_e^- - d_e^+ = \sum_i A_i \end{cases}$

Any additional hedging consiraints

Figure 2. Mathematical Formulations of BRIK

objective function. This also simplified programming. Finally, the third advantage gained from G.P. is flexibility. By just varying the priorities of the deviational variables, many different questions can be answered concerning the formulation. This flexibility will be better understood given a description of each of the three objective functions.

The first objective function is designed to take the available arsenal and allocate it to meet the DE goals as best possible. It has four parts and can use as many as ten preemptive priorities. Part one bounds the problem with the available arsenal. At priority one, P1, the positive deviational variables in the weapon constraints are minimized. This will prevent any lower goal from using more weapons than

are available. In the second part, all of the user-defined hedging options are met at priority two, P2. With G.P., if a particular goal is not completely satisfied, one of the two deviational variables will have some positive value. This fact makes it easy to determine if there is an insufficient arsenal to meet the hedges. Should there be an insufficient arsenal, some deviational variables in the hedging constraints will have a positive value in the solution. In part three of objective function one, weapons are allocated to the target classes in order of priority. BRIK permits division of the target classes into as many as seven preemptive priorities. This is done by choice of the user. If it is chosen not to separate the classes into priorities, all of the target goals will appear at priority three, P3.

At this point, it is convenient to discuss the weights put on the deviational variables for the targeting goals. As mentioned earlier, the use of weights permits ranking of the variables at each priority. This is normally accomplished by multiplying the deviational variable in the objective function by some value associated with a particular goal. However, this process has been complicated by the linearization of the constraints (see Chapter IV). Before the linearization, a target constraint had the following form:

$$\prod_i S_{ij}^{X_{ij}+n_j-p_j} = PV_j.$$

As mentioned earlier, minimizing p_j would drive the constraint to the survival goal of PV_j. After linearizing this

constraint, it was shown that

$$N_j \ln(1+\frac{p_j}{PV_j})$$

can play the role of the negative deviational variable d_j^-. Therefore, to drive the linearized constraint to its goal, it is necessary to now minimize d_j^-, because this term holds the variable p_j. If the constraints had not been linearized, the targets could be weighted within each priority by multiplying the deviational variable p_j by a number representing target class value (V_jN_j). However, the deviational variable in each constraint is not p_j, it is

$$d_j^- = N_j \ln(1+\frac{p_j}{PV_j}).$$

If this equality was solved for p_j, the result would be the following:

$$p_j = PV_j(e^{(d_j^-/N_j)}-1).$$

Multiplying both sides by V_jN_j yields

$$V_jN_jp_j = V_jN_jPV_j(e^{(d_j^-/N_j)}-1). \qquad (1)$$

Therefore, since d_j^- appears in each target constraint, to properly weight the deviational variables, the right side of equation one should be in the objective function with the goal of minimizing d_j^-. This is not solvable with LP! Therefore, the following approximation was used in BRIK. As

$$d_j^-/N_j$$

gets small,

$$(e^{(d_j^-/N_j)}-1) \approx d_j^-/N_j$$

yielding

$$V_jN_jp_j \approx V_jPV_jd_j^-.$$

Thus, the number used to weight the negative deviational variables in the target constraint in objective function one is

$$W_j = V_j PV_j.$$

The last part of objective one and the lowest priority goal is the extreme goal. The purpose of this goal is to minimize some aspect of the allocation. Because of the preemptive features of G.P., this goal may or may not be considered. If some of the DE goals cannot be met due to an insufficient arsenal, the extreme goal will not be considered. However, if all of the goals are met, there will be weapons left over. If that is the case, because of the use of non-integer programming, virtually an infinite combination of allocations could be reported. Thus, the extreme goal serves the role of driving the allocation to a unique solution. This first objective function performs an allocation restricted to weapon availability. Thus, depending on the size of the arsenal, the goals may or may not be met. This contrasts with the second objective function where the DE goals will be met.

The second objective function gives the user the ability to build an arsenal to meet all DE goals. This is accomplished with five preemptive priorities. At P1, the positive deviational variable in each target constraint is minimized, guaranteeing that the targeting goals will not be exceeded. At P2, all of the hedging constraints are met and at P3 all of the targeting goals are met. At this point, the user has

two options concerning the weapon base. The weapon classes can be input where the availability is zero or some number can be input that represents the "minimum" numbers of each weapon type that should be used in the allocation. Since weapon availability is unbounded, there is no limit to the number of weapons from each class that can be allocated. So, regardless of how many weapons the user inputs for the minimum arsenal, all DE goals will be met. However, should some minimum arsenal be entered, this objective function will attempt to use all of those weapons prior to "building" any additional warheads. This is accomplished at P4 by minimizing the negative deviational variables on the weapon constraints. There is a side-effect for this priority of which the user must be aware. Should the minimum arsenal be sufficient to meet all of the DE goals, P4 will have the effect of maximizing the number of warheads used and the following priority, P5, will have no effect. Finally, just like the lowest priority goal in the first objective function, P5 insures some minimum aspect of the used arsenal.

The final objective function, type three, has four parts and requires four preemptive priorities. Unlike the first two objectives which bound the problem with either weapon availability or DE goals, this third objective function bounds the problem with both. At P1, the positive deviational variables of the weapon constraints are minimized. Also, at P2, the positive deviational variables of the target

constraints are minimized. This guarantees that neither the weapon availabilities nor the designated goals will be exceeded. At P3, all of the hedging options will be met within the limits set by P1 and P2. Finally, the extreme goal will be considered at P4. If this objective function is chosen, there are two major differences from the first two objective functions of which the user must be aware. First, the DE goals are not used to drive the allocation. They are used only as upper bounds to the problem. Second, the extreme goal has a fifth option that is not present in the other two objectives. That option permits dumping any remaining arsenal on targets whose goals are not yet met. Finally, the type three objective function relies heavily on the hedging constraints to drive the allocation. Therefore, it is estimated that it would take longer to set up a problem using the type three function as opposed to the type one or the type two objective.

In conclusion, G.P. provides a tremendous amount of flexibility. So much in fact, it could be argued that restricting the model to only three objective functions restricts some of the gain from G.P. It could have been possible to build BRIK such that the user could stipulate what variables from what constraints go into what priorities. That indeed would have permitted the user to take full advantage of the mathematics. However, it was decided not to allow the ability to manually juggle objective terms because of our stated purpose of

developing a user-friendly model. A user does not require an in-depth knowledge of G.P. to use BRIK. To select one of the above objective functions, all that is needed is the ability to choose between 1, 2, or 3. Everything is then built internally.

G.P. has been described and the formulations have been presented. The last part of this chapter concerns the algorithm used to solve the problem just described. That algorithm is called a Partitioning Algorithm for Goal Programming Problems (PAGP).

PAGP

BRIK's evolution into a G.P. model occurred simultaneously with the search for an independent allocation routine. To increase the model's portability, it was concluded early in the project that BRIK would not be fettered by any local library package. At first, time was spent looking for an LP package that could be used as a subroutine. But as the current model evolved, the search expanded to include goal programming routines. While it was not beyond the scope of this effort to write a G.P. subroutine, finding an already existing package helped simplify the coding problem. It was decided to use PAGP because, besides meeting all of the requirements for portability, the program had coding efficiencies of which BRIK could take advantage. These efficiencies include:

1) Partitioning of the goal constraints.

2) Variable elimination.

3) Special termination rules.

This section will describe each of these three coding "tricks" included in PAGP and explain why they are an advantage to BRIK.

The first advantage of PAGP is the partitioning procedure. The ordinal priority factors in the goal programming objective function are used to partition the goal constraints of the problem. "This is accomplished first by observing that for any goal constraint i, one and only one of three things may occur:

1) only d_i^- appears in the objective function,

2) only d_i^+ appears in the objective function, or

3) both d_i^- and d_i^+ appear in the objective function.

In case (1), the partition would assign goal constraint i to the priority factor associated with d_i^-; in case (2), constraint i would be assigned to the priority factor associated with d_i^+; while in case (3), the partition would determine the higher order priority factor (in terms of the ordinal ranking) associated with either d_i^- or d_i^+ and constraint i would be assigned to that priority" (Ref 2:379). Only the constraints that had deviational variables in P1 are in the initial problem. Next, the only new constraints added at P2 are those constraints that had deviational variables at P2 and not at P1. This continues until either all of the constraints in the problem are added or the stopping rule is

invoked. BRIK takes advantage of partitioning because different sets of constraints are brought in at different priorities. Consider objective one. Only the weapon constraints are in the problem at P1, the hedging constraints are brought in at P2, the target constraints are added at P3, and finally the extreme goal is input at P4. PAGP is actually solving a series of smaller subproblems in order to find a solution to the original problem. This increases efficiency.

The second "trick" in PAGP is the elimination procedure. Not only does it help BRIK, it improves efficiency in any problem solved by PAGP. The motivation behind the elimination procedure comes from the theory of L.P. If Z is the optimal value of an L.P. problem and $(z_j-c_j)>0$ for some nonbasic x_j, x_j cannot enter the basis to form an alternative optimal solution. A corollary to this statement that applies to G.P. concerns the optimal solution to a subproblem S_k. Any nonbasic variable which has at least one positive relative cost $(z_j-c_j>0)$ can be eliminated from entering the basis in subproblem S_y, where $y = k+1,\ldots,P$ (Ref 2:380). This elimination makes the program more efficient because fewer nonbasic elements are considered to enter the basis at each pivot. PAGP accomplishes this by maintaining an indicator row, IND. While this indicator row increases the efficiency of PAGP, it should be noted that PAGP still pivots all columns. The program could be further improved if coded not to pivot the eliminated columns.

The third "trick" used in PAGP is a special stopping rule. This is accomplished by using the vector IND. If it indicates no nonbasic elements can enter the basis, the routine stops, regardless of how many lower priorities are left to be considered. BRIK's extreme goal takes advantage of this last efficiency item. In objective one, the DE goals are met as best possible at P3. At P4, if all of the DE goals are attained, some aspect of the allocation is minimized. However, if some DE goals were not met, PAGP's stopping rule prevents the constraint at P4 from ever being added to the G.P. matrix. This results in added efficiency.

Conclusion

This concludes the mathematical presentation of BRIK. It is a goal programming model that has its own independent allocation routine. The User Guide is in Appendix A and a listing appears in Appendix E. The following chapter concerns verification and demonstration.

V. Verification and Demostration

This chapter will discuss verification of BRIK, then demonstrate some of its capability. As defined by the Random House Dictionary of the English Language, to verify means to "ascertain correctness of, as by examination, research, or comparison." This section examines the steps taken to verify the correctness of this model. The demonstration section will start with a sample problem that compares BRIK solutions with output from AEM. Then, sample results used to illustrate BRIK's objective functions and hedging options will be displayed.

Verification

BRIK has 3725 lines of code divided among 49 subroutines. This section will describe the effort undertaken to ascertain its correctness. Also, this section indirectly provides a more detailed description of the physical composition of the model by reviewing the functions of the subroutines and the use of the data files that are built. While most small algorithms can be verified with hand calculations, verification of a program of this size must use some type of systematic approach. The approach used divided the model into six sections:

1) Editor
2) SSPS Computation
3) Matrix Formulation
4) Objective Development

5) PAGP (Assignment Algorithm)

6) Output

The steps taken to insure verification of each section will be presented separately.

Editor. The editor has two parts, the target editor and the weapon editor. Except for the fact that one is for target data and the other is for weapon data, they are essentially the same. The verification process consisted of exercising the program to determine if both sections were acting as expected. For example, both methods of data entry, either through files or interactively, were exercised until it was visually determined that information was going into the proper storage locations. This visual capability was made possible by the subroutines TGTEDT and WPNEDT. Each of these subroutines permit a viewing of the data on the monitor.

After it was determined that weapon and target data was being entered properly, effort was concentrated on the transportation between subroutines called by TGTEDT and WPNEDT. When outlining the flowchart for this model, it was determined that it was easiest to control transportation if the user was returned to a main menu after executing a particular option. This technique proved invaluable, especially since the first option on each menu permitted the user to view the data base. Verification again consisted of exercising the program. After performing the different menu options, a visual inspection of the data was made to insure

that the desired results were attained. Eventually, it was possible to move throughout both editors and always return to their respective main menu. That eventual return coupled with a correct data display insured validity of the editors.

SSPS. When BRIK was in its earliest stages of development, it was visualized to have two separate programs. The first program would contain the editor and the matrix generator and the second program would have the allocation routine and the output. The intention was to parallel the BLUE SHADE model at SAC Headquarters. To facilitate these two programs, files were built by the first program that would be used as input for the second program. One of those files is called SPARSE.

SPARSE is a j by i matrix containing the single shot probability of survival (SSPS) of a target in class j if assigned a warhead from a weapon in class i. There are two different sources for each SSPS number calculated. If the user inputs the target vulnerability in terms of PSI, probability of kill (PK) is calculated by function PK. If a VNTK number is input to express target vulnerability, PK is computed by Function VTK. After a PK is computed, SSPS is attained by subtracting the product of PK and weapon reliability from one. Each of the two functions, PK and VTK, were verified separately.

Function PK was verified by following a series of inputs with hand calculations. These calculations matched the entries of SPARSE. Therefore, PK was assumed to be correct.

This relatively simple process contrasts greatly with the steps taken to verify Function VTK.

VTK is a function extracted directly from BALLOT. Since BALLOT has enjoyed extensive use at SAC Headquarters, it was assumed that the output would be acceptable if the function was being used properly. First, it was necessary to determine the units of the input variables. This was gleaned from the program listing. In BALLOT, CEP is in feet, Yield is in kilotons, and R-95 is in nautical miles. Second, sample calculations were made with the function to insure reasonableness. When the authors were satisfied that the function was being properly used, tables similar to Figures 3-A and 3-B were computed for both T factors, P and Q.

The intuitive consistency of these tables is sufficient assurance of the verification of Function VTK. In the center row, as yield increases, the PK increases. In the center column, as CEP increases, PK decreases. Both of these make sense. Also, notice that the PK increases as K increases. The K factor allows for hardness adjustments to be made to account for the effects of variations in blast wave duration due to different weapon yields (Ref 5:34). Since targets with a high K factor are more susceptible to damage from shock wave duration than targets with a low K factor, it makes sense that as the K factor increases, so does PK.

To further verify Function VTK, a comparison was made between PK's computed by BRIK and some PK's computed by AEM.

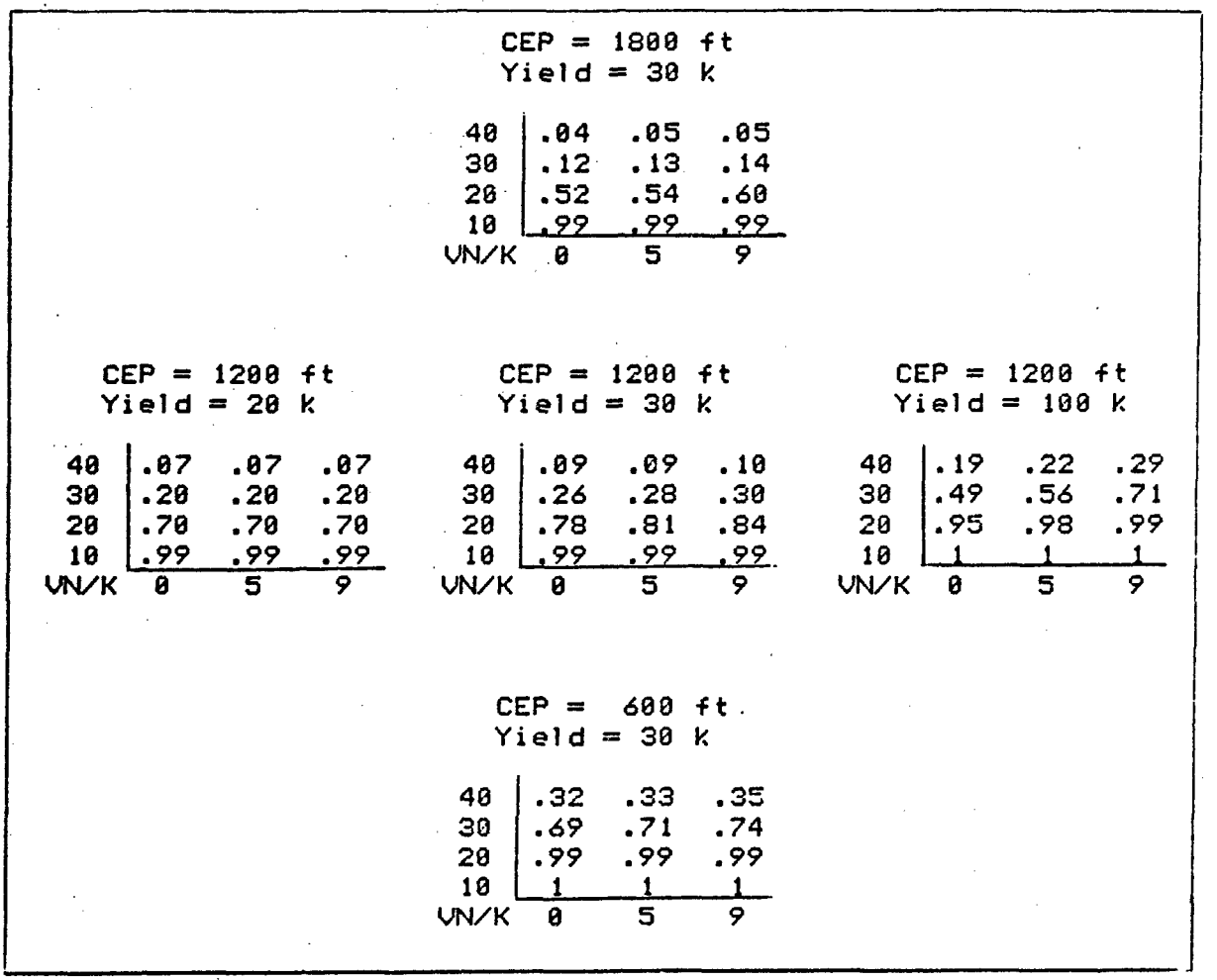


Figure 3-A. Various Probability of Kill
(R 95 = 0 and T = P)

The first sample problem presented in the demonstration section of this chapter contains five weapon classes and ten target classes. To solve this problem, it was necessary for both programs to compute 50 different PK's, one for each weapon/target combination. A visual comparison of those two sets of numbers revealed that Function VTK's numbers matched AEM's probability of kill computations to the third, and

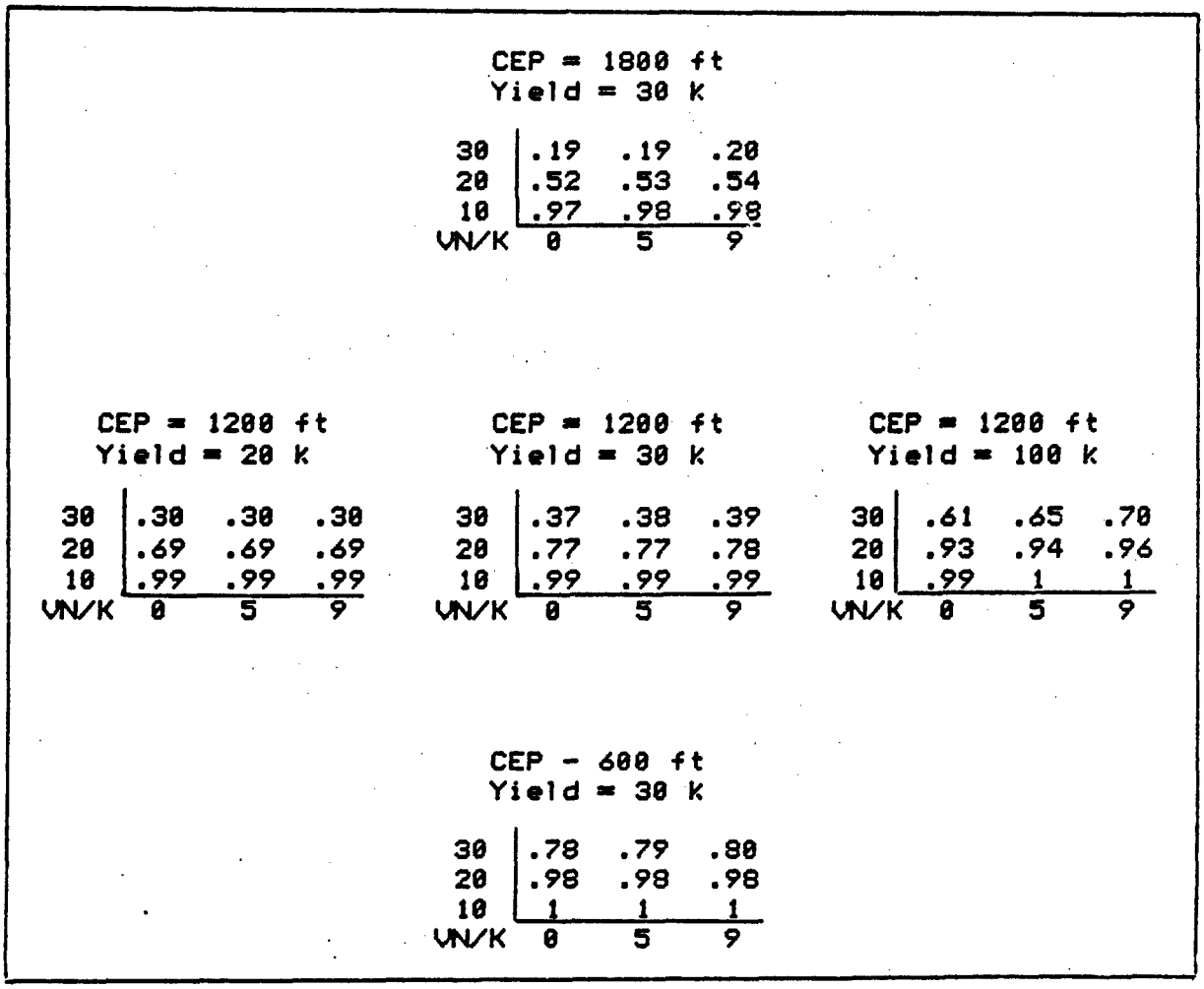


Figure 3-B. Various Probability of Kill (R 95 = 0 and T = Q)

sometimes the fourth decimal place. This further verifies that the borrowed subroutine is being used correctly.

In conclusion, verification of the SSPS computations for PSI inputs came from hand calculations and verification of the VNTK system calculations came from a comparison with an established nuclear exchange model.

Matrix Formulation. AIJ is a file that contains the matrix of the current problem. This file was built primarily

to provide the output section right-hand-side values. Its secondary purpose was to simplify the verification process. Small problems were run using all of the hedges and each of the five types of extreme goals. The AIJ files generated for each run were examined for verification.

There were two examinations the files had to pass, the numbers in each constraint had to be in the proper columns and they had to be accurate. A visual inspection verified that the numbers were in the right columns for the small sample problems. Since the numbers are properly located for small problems, they will be properly positioned for large problems because the algorithm which is used for column placement--

$$[1 + j + (i-1)*NTGTS]$$

where i is the weapon class number, j is the target class number, and NTGTS is the number of target classes in the current problem--is dependent on problem size. After verifying that the numbers were in their correct positio s, it was necessary to check their correctness. The coef-ficients in the target constraints are the negative loga-rithms of the SSPS's found in SPARSE. It was a simple matter to verify those numbers having both SPARSE and AIJ available. Right hand sides, EMT, and CMP figures were all checked with hand calculations.

Objective Development. PAGPIN is a file used by the goal progamming package. It contains all of the information PAGP

requires to solve the goal programming problem. Since all of the data describing the objective functions is in this file, verification consisted of visually checking its accuracy. It must be noted that the information contained in PAGPIN is the authors' interpretation of PAGP's requirements. This interpretation resulted from reading the program listing acquired from ACM. The verification of that interpretation will be discussed in the next section.

PAGP. Two things were required to verify PAGP, verification of the algorithm and verification of the interpretation of its requirements. It was decided that this could be accomplished by using it to run text book examples. Despite PAGP being proofed prior to publication and the test examples coming from a text book, it is presumptuous to think that both could be without error. However, because they are independent, the chance of achieving the same answer given that one or both are in error is considered negligible. Thus, the examples on pages 394, 404, and 408 from Ignizio (Ref 18) were run. PAGP computed the same answers reported in the text. Therefore, since the output compared favorably, three statements could be made:

1) The program PAGP was operating properly.

2) The text book examples were correct.

3) Our interpretation of the needs of PAGP was accurate.

Output. The output routine uses the results from the allocation and computes the damage achievement. This

achievement is displayed on the monitor screen along with the target classes attacked, the weapon systems that attacked them, the number of weapons assigned to each target class, and the expected number of targets remaining in each target class. Since the output section uses the number of weapons allocated to compute damage achievement, verifying the accuracy of the achievement would insure verification of the output section.

Verification of the achievement is as follows. At this point, PAGP is working properly. Therefore, if the problem is formulated correctly and there are sufficient weapons in the inventory to meet the DE goals, the reported damage achievement from an allocation should equal the goals. Example problems were run and the achievement did equal the goals. Therefore, the output section is verified.

This concludes the formal process of verifying BRIK. It is allocating weapons to targets exactly as it is being asked.

Demonstration

This demonstration section consists of two parts. In the first part, a test problem is presented where a solution generated by AEM is compared to solutions from BRIK. The second part runs through a series of small examples to show the capability and versatility of BRIK.

Test Problem. BRIK allocates weapons to targets only if the DE goals are known. Therefore, to compare BRIK with the Arsenal Exchange Model, the following problem was designed. A target base and a weapon base were supplied by the authors to

USAF/SA. These two bases were run on AEM with the objective to maximize DE. There were no special allocation rules or hedges invoked in the problem. After receiving AEM's allocation, the DE attained on each target class became the DE goals for BRIK. This made it possible to form a comparison between the two models. The following is an in-depth description of the entire process along with the results.

The problem is to allocate a weapon base, consisting of five classes, to a target base consisting of ten classes (Table VI) in order to maximize damage expectancy (DE).

Table VI

Test Problem

10

NAME	NUMB	VUL	DIA	TYPE	PRIO	VAL	REL	CEP	YLD	WHDS
civil	140	24Q0	.56	m	1	1.00	.00	.00	.00	0
local	215	13P1	.49	m	1	1.00	.00	.00	.00	0
c3i	450	35P7	.00	m	1	1.00	.00	.00	.00	0
icbm	1000	52P8	.00	f	1	1.00	.00	.00	.00	0
lcc	200	39P0	.00	f	1	1.00	.00	.00	.00	0
subpts	20	22P1	.36	f	1	1.00	.00	.00	.00	0
irbm	150	11P0	.00	f	1	1.00	.00	.00	.00	0
afbase	100	10Q1	.79	f	1	1.00	.00	.00	.00	0
stores	430	31P6	.00	m	1	1.00	.00	.00	.00	0
facil	520	23Q0	.31	m	1	1.00	.00	.00	.00	0

5

NAME	NUMWPNS	WHD/WPN	REL	CEP	YLD	DAYALRT	GENALRT
MMII	450	1	.85	.200	2.00	.85	1.00
MMIII1	250	3	.85	.150	.17	.85	1.00
Poseid	104	10	.85	.240	.05	.85	1.00
B52grv	106	4	.85	.600	.35	.33	1.00
FB111	60	6	.85	.200	.20	.33	1.00

The first half of Table VI consists of the number of target classes in the base, the header row, and ten rows that fully describe each target class. Column one, NAME, gives the name of each class, and column two, NUMB, gives the number of targets in each class. Column three, VUL, is the target vulnerability index. In this problem VNTK numbers are used. Column four, DIA, gives the diameter of the smallest circle that encompasses 95% of the target. Columns five and six, TYPE and PRIO, indicate the type of target class and each class' priority. In this problem all target classes are priority 1. Finally, column seven, VAL, shows the value of each target in the class. All the targets in this problem have a value of one. The last four columns, REL, CEP, YLD, and WHDS are used to calculate "force target" values and are not used in this problem. Therefore, the input numbers are zero.

The next section of Table VI contains the weapon base. The first two rows are similar to the target base. They contain the number of weapon classes in the base and the header row. For this problem, there are five weapon classes. Each weapon class name and the number of weapon systems available in each class are listed in columns one and two, NAME and NUMWPNS. The third column, WHD/WPN, indicates the number of warheads each weapon system contains. The fourth column, REL, is the probability that each warhead will arrive and detonate, while the fifth column is the weapon system CEP

in nautical miles. The sixth column, YLD, is the yield in megatons of each warhead. The final two columns, DAYALRT and GENALRT, contain the percentage of weapon systems on alert.

AEM's solution is given in Table VII. The target class name is listed first. The next two columns, value and type, are self-explanatory. The fourth column, no., is the number of targets in the class. Column six is the number of targets that survived the allocation. Columns seven and eight are not used. Column nine provides the expected percentage of kill in each target class. If more than one weapon type attacked the target class, the total expected kill percentage is in the row marked "sum". Columns 10, 11, and 12 give the allocation information. For example, with target class "civil" in the top row, 20 targets (column 10) were each attacked by 1 warhead (column 11) of weapon type fb111 (column 12). Although AEM's total value destroyed is not given in the table, AEM's score was 1770.

Since BRIK allocates weapons to targets to meet DE goals, the DE achieved by AEM for each target class was used as input for BRIK. Objective function one and extreme goal one (meet goals with the fewest amount of weapons) were used in the allocation. The output is listed in Table VIII. The first two columns list the target class name and the weapon types allocated to it. The third column is the number of weapons of the weapon class allocated to the target class. The fourth column is the DE goal. The fifth column is the percentage of

TABLE VII

Test Problem AEM Solution

target	value	type	no.	survivors	max kill	min kill	expected kill	number attacked	strategy warheads	by weapon
1	2	3	4	5	6	7	8	9	10	11	12
1 civil	1.000	2.	140.	33.20	1.00	0.00	0.729	20.00	1. type	5(fb111
							0.769	120.00	1. type	2(mm3-1
						sum ---	0.763	140.00		
2 i_{ccal}	1.000	2.	215.	5.47	1.00	0.00	0.975	215.00	2. type	3(poseid
3 c_{3i}	1.000	2.	450.	120.14	1.00	0.00	0.832	250.00	1. type	1(mm2
							0.609	200.00	1. type	2(mm3-1
						sum ---	0.733	450.00		
4 i_{cbm}	1.000	1.	1000.	1000.00	1.00	0.00				
5 i_{cc}	1.000	1.	200.	65.65	1.00	0.00	0.672	200.00	1. type	1(mm2
6 a_{ubpts}	1.000	1.	20.	3.97	1.00	0.00	0.801	20.00	1. type	5(fb111
7 i_{rbm}	1.000	1.	150.	8.43	1.00	0.00	0.850	38.00	1. type	3(poseid
							0.976	112.00	2. type	4(b52-grav
						sum ---	0.944	150.00		
8 a_{fbase}	1.000	1.	100.	2.69	1.00	0.00	0.973	100.00	2. type	4(b52-grav
9 stores	1.000	2.	430.	121.09	1.00	0.00	0.718	430.00	1. type	2(mm3-1
10 f_{acil}	1.000	2.	520.	94.36	1.00	0.00	0.890	172.00	3. type	3(poseid
							0.784	320.00	1. type	5(fb111
							0.771	28.00	2. type	3(poseid
						sum ---	0.819	520.00		

TABLE VIII

Test Problem BRIK Solution: AEM Achievement

As BRIK Goals

****

TOTAL VALUE DESTROYED WAS 1770.2.

TGTNAM	WPNCLASS	NUMBER ASSIGNED	GOAL	ACHIEVEMENT	NUMBER REMAINING
civil	FB111	154.93	.76	.76	33.2
local	Poseid	430.36	.98	.98	5.4
c3i	MMII	250.09	.73	.73	120.2
	MMIII1	156.21			
icbm			.00	.00	1000.0
lcc	MMII	199.91	.67	.67	65.6
subpts	FB111	20.05	.80	.80	4.0
irbm	Poseid	167.51	.94	.94	8.4
	B52grv	61.58			
afbase	B52grv	199.26	.97	.97	2.7
stores	MMIII1	430.21	.72	.72	121.3
facil	MMIII1	163.57	.82	.82	94.1
	Poseid	442.13			
	FB111	185.02			

WPNAME	NUMBER	USED	REMAIN
MMII	450.0	450.0	.0
MMIII1	750.0	750.0	.0
Poseid	1040.0	1040.0	.0
B52grv	424.0	260.8	163.2
FB111	360.0	360.0	.0

the target class destroyed in the allocation. It is not the percentage of the goal met! The final column is the number of targets remaining after the allocation.

There are many similarities between the two allocations. The total value destroyed was the same and many of the target classes received the same type and amount of weapon types. However, there are two discrepancies. BRIK indicated that there were unused weapons. Also, four target classes were not assigned the same type or number of weapons. Because of these discrepancies, several iterative runs were made, gradually increasing the DE goals until the remaining weapons were used. The final result is listed in Table IX.

The most notable exception between BRIK's final run in Table IX and AEM's solution is the increase in value destroyed, 1788. This is due to BRIK reporting a slightly higher damage achievement than AEM for some of the target classes. For example, BRIK reports a DE of .75 for "C3i" while AEM reported .733. This higher DE occurs approximately four times in BRIK's answer and is caused by the damage function approximation described in Chapter III.

Even though BRIK's output does not match AEM's solution exactly, they are very close. Using AEM's solution as the DE goals, a difference of only 6% in the amount of weapon usage occurred. When BRIK's DE goals were gradually increased until all of the weapons were used, a difference of 1% in the value destroyed was attained. In an attempt to determine why the

TABLE IX

Test Problem BRIK Solution

****

TOTAL VALUE DESTROYED WAS 1788.4.

TGTNAM	WPNCLASS	NUMBER ASSIGNED	GOAL	ACHIEVEMENT	NUMBER REMAINING
civil	MMIII1	122.56	.77	.77	32.2
	FB111	20.86			
local	Poseid	428.04	.98	.97	5.5
c3i	MMII	250.09	.75	.75	110.3
	MMIII1	197.23			
icbm			.00	.00	1000.0
lcc	MMII	199.91	.67	.67	65.6
subpts	FB111	20.05	.80	.80	4.0
irbm	Poseid	24.40	.95	.95	6.8
	B52grv	224.74			
afbase	B52grv	199.26	.97	.97	2.7
stores	MMIII1	430.21	.72	.72	121.3
facil	Poseid	587.56	.83	.83	88.4
	FB111	319.09			

WPNAME	NUMBER	USED	REMAIN
MMII	450.0	450.0	.0
MMIII1	750.0	750.0	.0
Poseid	1040.0	1040.0	.0
B52grv	424.0	424.0	.0
FB111	360.0	360.0	.0

differences occurred, the single shot probabilities of survival as computed by BRIK were compared with those of AEM. As mentioned in the verification section, these numbers compared favorably. It was determined that the differences in destroyed value were attributed to the damage function approximation described in Chapter III. This approximation "error" will be further described in Chapter VII.

BRIK Capabilities. BRIK is not a single use model. Instead, due to the various objective functions and constraining rules that can be combined for use in any problem, BRIK has the versatility to be used in many types of nuclear exchange analyses. This section will demonstrate the use and interaction of BRIK's objective functions and hedges in the allocation process.

Examples will be used to clarify the use of the various rules that can be employed in BRIK. These examples will be small and simple, so that the effect of each function or constraint is not hidden in the allocation. A target base (Figure 4-A) of two classes will be attacked by a weapon base (Figure 4-B) of two classes for all of this section's examples. The DE goals for both target classes will be .95 with enough weapons available to meet these DE goals unless specified otherwise. The extreme goal used in all problems was to minimize warheads used. The first examples will demonstrate the three objective functions. The remaining examples will describe the effect of each hedge.

2										
NAME	NUMB	VUL	DIA	TYPE	PRIO	VAL	REL	CEP	YLD	WHDS
civil	10	25QB	.49	m	1	1.00	.00	.00	.00	0
local	10	10Q1	.49	m	1	1.00	.00	.00	.00	0

Figure 4-A. Example Target Base

2							
NAME	NUMWPNS	WHD/WPN	REL	CEP	YLD	DAYALRT	GENALRT
titan	49	1	.80	.900	4.00	.85	1.00
MMII	45	1	.85	.300	1.00	.85	1.00

Figure 4-B. Example Weapon Base

Objective Functions: There are three objective functions in BRIK from which to choose. They include the following:

1. DE goals are met as best possible using only the available arsenal.
2. Force all DE goals to be met, regardless of the weapon availability.
3. Bounds the problem with both weapon availability and DE goal levels. This function requires hedging constraints or a number 5 extreme goal--forcing BRIK to use all remaining weapons--to run an allocation.

The following sample output (Figure 5) is from an allocation with a type one objective function. Both target DE goals were met using 33.4 MMIIs. The remaining MMIIs and Titans were not allocated and were placed in reserve.

TOTAL VALUE DESTROYED WAS 19.0.

TGTNAM	WPNCLASS	NUMBER ASSIGNED	GOAL	ACHIEVEMENT	NUMBER REMAINING
civil	MMII	17.65	.95	.95	.5
local	MMII	15.79	.95	.95	.5

WPNAME	NUMBER	USED	REMAIN
titan	49.0	.0	49.0
MMII	45.0	33.4	11.6

Figure 5. Objective One Example Output

The second objective function forces all DE goals to be met and, if necessary, creates weapons to meet the DE goals. The function will either build the weapon classes and numbers needed from scratch or it can build additional weapons in addition to those already included in the weapon base. Only weapon classes already in the weapon base will be used in the allocation. The following example (Figure 6) is run with both weapon classes having 0 weapons. In order to meet its goals, BRIK had to create 33.4 MMIIs, which is the same amount needed in the first objective example. The weapon usage output indicated that -33.4 MMIIs were remaining. This is interpreted as adding 33.4 MMIIs to the existing arsenal.

If there are already weapons in the inventory, BRIK will use those weapons already existing before adding more to the weapon base. For example, for the solution reported in Figure 7, instead of 0 weapons in both classes, the Titan class initially had ten weapons and the MMII class had 0. The allocation used all the 10 Titans and added 25 MMIIs to meet the goals.

The type two objective function was specifically designed to use whatever minimum weapons were made available prior to "building" more. If the user provides a sufficient arsenal to meet the DE goals and uses the type 2 objective function, BRIK will allocate the maximum number of weapons possible to meet those goals.

TOTAL VALUE DESTROYED WAS 19.0.

TGTNAM	WPNCLASS	NUMBER ASSIGNED	GOAL	ACHIEVEMENT	NUMBER REMAINING
civil	MMII	17.65	.95	.95	.5
local	MMII	15.79	.95	.95	.5

WPNAME	NUMBER	USED	REMAIN
titan	.0	.0	.0
MMII	.0	33.4	-33.4

Figure 6. Objective Two Example Output - 0 weapons

TOTAL VALUE DESTROYED WAS 19.0.

TGTNAM	WPNCLASS	NUMBER ASSIGNED	GOAL	ACHIEVEMENT	NUMBER REMAINING
civil	MMII	17.65	.95	.95	.5
local	titan	10.00	.95	.95	.5
	MMII	7.31			

WPNAME	NUMBER	USED	REMAIN
titan	10.0	10.0	.0
MMII	.0	25.0	-25.0

Figure 7. Objective Two Example Output - 10 Titans

The next example is an allocation using a type three objective function and no hedging or type 5 extreme goal constraints. The allocation places an upper limit on the DE that can be accomplished and the weapons that can be used. The type 1 extreme goal used minimizes the weapons. Therefore, since there is no minimum DE level specified and the extreme goal forces a minimization of weapons, none of the weapons are used (Figure 8). If a hedge is added to force a .5 DE on the target class "civil", BRIK will allocate 4.1 MMII weapons to achieve the DE level (Figure 9). It should be noted that BRIK used 4.1 weapons to destroy 5 targets; this is due to the same damage function assumption which was evident in the AEM comparison in the previous section. This problem is accentuated in small problems, and is not as evident in larger, more complex problems.

TOTAL VALUE DESTROYED WAS .0.

TGTNAM	WPNCLASS	NUMBER ASSIGNED	GOAL	ACHIEVEMENT	NUMBER REMAINING
civil			.95	.00	10.0
local			.95	.00	10.0

WPNAME	NUMBER	USED	REMAIN
titan	49.0	.0	49.0
MMII	45.0	.0	45.0

Figure 8. Objective Three Example Output - No Hedges

TOTAL VALUE DESTROYED WAS 5.0.

TGTNAM	WPNCLASS	NUMBER ASSIGNED	GOAL	ACHIEVEMENT	NUMBER REMAINING
civil	MMII	4.08	.95	.50	5.0
local			.95	.00	10.0

WPNAME	NUMBER	USED	REMAIN
titan	49.0	.0	49.0
MMII	45.0	4.1	40.9

Figure 9. Objective Three Example Output - l'edge

Another way to force a type three objective function to allocate, is to include a type 5 extreme goal to force BRIK to use all of the remaining weapons. To demonstrate this method, both weapon classes were reduced to 10 weapons each and the

results are displayed in Figure 10. The extreme goal forced BRIK to allocate all 20 weapons.

TOTAL VALUE DESTROYED WAS 13.7.

TGTNAM	WPNCLASS	NUMBER ASSIGNED	GOAL	ACHIEVEMENT	NUMBER REMAINING
civil	titan	10.00	.95	.87	1.3
	MMII	6.35			
local	MMII	3.65	.95	.50	5.0

WPNAME	NUMBER	USED	REMAIN
titan	10.0	10.0	.0
MMII	10.0	10.0	.0

Figure 10. Objective Three Example Output - Type 5 Extreme Goal

Hedges: The following seven constraining hedges are available in BRIK:

1. Enforce a minimum level of damage on a particular target class.
2. Enforce a minimum level of damage on a particular target class with a specific subset of weapon classes.
3. Allow only a certain amount of damage (%) on a particular target class resulting from a specific set of weapons.
4. Restrict the number of weapons from a weapon class that can be allocated to a specific target class.
5. Allows the construction of a constraint that is not indigenous to BRIK.

6. Enforce a minimum level of damage on a particular aggregated set of target classes.
7. Restrict the amount of total weapons that can be allocated to each target in a specific target class.

If there is a certain minimum level of damage that must be attained on a target class, a type one hedging constraint can be used. This hedge was used in Figure 8 to force the third objective function to allocate. If it was desired that the DE level be accomplished using Titans instead of MMIIs, a type two hedge could be used. This would allow the user to specifically designate the titan class as the only weapon class to be allocated to target class "civil" to attain the specified DE level (Figure 11).

TOTAL VALUE DESTROYED WAS 5.0.

TGTNAM	WPNCLASS	NUMBER ASSIGNED	GOAL	ACHIEVEMENT	NUMBER REMAINING
civil	titan	7.27	.95	.50	5.0
local			.95	.00	10.0

WPNAME	NUMBER	USED	REMAIN
titan	49.0	7.3	41.7
MMII	45.0	.0	45.0

Figure 11. Example of Type Two Hedging

To demonstrate a type 3 hedge, consider the following. Figure 4 showed output when an objective function one was used to attain .95 DE levels. If it is desired to restrict the MMIIs from inflicting greater than 50% of the DE on the "civil" class, a type three hedging constraint can be used. This hedge allowed BRIK to use only 4.08 MMIIs to attain a .5 DE of the target class (Figure 12). Titans are used to meet the rest of the DE goal.

TOTAL VALUE DESTROYED WAS 19.0.

TGTNAM	WPNCLASS	NUMBER ASSIGNED	GOAL	ACHIEVEMENT	NUMBER REMAINING
civil	titan	24.16	.95	.95	.5
	MMII	4.08			
local	MMII	15.79	.95	.95	.5

WPNAME	NUMBER	USED	REMAIN
titan	49.0	24.2	24.8
MMII	45.0	19.9	25.1

Figure 12. Example of Type Three Hedging - Single Weapon Class

If the weapon subset included both "titan" and "MMII" classes, BRIK will only attain a DE level of .5 on the "civil" class (Figure 13).

TOTAL VALUE DESTROYED WAS 14.5.

TGTNAM	WPNCLASS	NUMBER ASSIGNED	GOAL	ACHIEVEMENT	NUMBER REMAINING
civil	MMII	4.08	.95	.50	5.0
local	MMII	15.79	.95	.95	.5

WPNAME	NUMBER	USED	REMAIN
titan	49.0	.0	49.0
MMII	45.0	19.9	25.1

Figure 13. Type Three Hedging - Both Weapon Classes

If it is desired to limit the number of a weapon class to be used against a specific target class, a type four hedging constraint can be added. Figure 14 lists the output for a type one objective function. A type four hedge was used to prevent more than 5 MMIIs from being used in attaining the .95 DE goal against target class "local".

TOTAL VALUE DESTROYED WAS 19.0.

TGTNAM	WPNCLASS	NUMBER ASSIGNED	GOAL	ACHIEVEMENT	NUMBER REMAINING
civil	MMII	17.65	.95	.95	.5
local	titan	12.72	.95	.95	.5
	MMII	5.00			

WPNAME	NUMBER	USED	REMAIN
titan	49.0	12.7	36.3
MMII	45.0	22.6	22.4

Figure 14. Example of a Type Four Hedge

Since BRIK's set of hedges is not inclusive of all the possible allocation rules that might be needed for a problem, the ability to input a user-built constraint is included in BRIK. This is a type 5 hedging constraint. One possible use of this hedge is to account for a common carrier. If a weapon system contains two different types of warheads for use in a specific ratio, such as a B-52 carrying gravity bombs and SRAMs, this function can be used to build a constraint that only allows use of weapon systems in a specific ratio. For example, if it is desired to use the same exact amount of titans as MMIIs, a hedge can be built to force the same use of both weapon classes. The output from an allocation which used a type one objective function and a user-built hedge is shown in Figure 15.

TOTAL VALUE DESTROYED WAS 19.0.

TGTNAM	WPNCLASS	NUMBER ASSIGNED	GOAL	ACHIEVEMENT	NUMBER REMAINING
civil	MMII	17.65	.95	.95	.5
local	titan	18.10	.95	.95	.5

WPNAME	NUMBER	USED	REMAIN
titan	49.0	18.1	30.9
MMII	45.0	18.1	26.9

Figure 15. Example of a User-Built Hedge - Type 5

In the screen output, BRIK does not report any individual weapon allocation less than .5 , but it does add up all of the weapons allocated to determine the usage. This describes why there is a dexcrepancy between the number of weapons allocated and the number of weapons that are reported used. An inspection of "alloc" indicated that .45 MMIIs were allocated against the target class "local".

Hedge 6 can be used to create a super target class from several smaller ones. There might be times when it is desired to attain a DE level against an entire category of classes and not be particularly interested in the DE on specific target classes, especially if the target class characteristics vary too much to allow aggregation into a single class. For this, a type six hedge is used. It is recommended that this hedge be used only with the type 3 objective function, because this function places an upper bound on the DE that can be attained on each class. Without this upper bound, BRIK could allocate all of its weapons to the softest target in the set. The results in Figure 16 are from an example that used a type three objective function and a type six hedge of .5 DE on a combination of both classes. There are 10 targets in each class for a total of 20 targets. However, BRIK allocated 7.31 MMIIs to destroy 7.5 "local" targets, which was below the desired 10 target destruction goal. This is due to the damage function approximation and will be discussed in Chapter VII.

TOTAL VALUE DESTROYED WAS 7.5.

TGTNAM	WPNCLASS	NUMBER ASSIGNED	GOAL	ACHIEVEMENT	NUMBER REMAINING
civil			.95	.00	10.0
local	MMII	7.31	.95	.75	2.5

WPNAME	NUMBER	USED	REMAIN
titan	49.0	.0	49.0
MMII	45.0	7.3	37.7

Figure 16. Example of a Type Six Hedge

A type seven hedge restricts the amount of weapons that can be used against each target in a target class. Figure 17 shows example output from an allocation that used a type one objective function and two type 7 constraints. The constraints forced a maximum of one weapon per target in each target class. There were ten targets in each class, therefore, only ten weapons were allocated to each class.

TOTAL VALUE DESTROYED WAS 16.7.

TGTNAM	WPNCLASS	NUMBER ASSIGNED	GOAL	ACHIEVEMENT	NUMBER REMAINING
civil	MMII	10.00	.95	.82	1.8
local	MMII	10.00	.95	.85	1.5

WPNAME	NUMBER	USED	REMAIN
titan	49.0	.0	49.0
MMII	45.0	20.0	25.0

Figure 17. Example of a Type Seven Hedge

This concludes the demonstration. BRIK incorporates several objective functions and many allocation rules into an extremely versatile nuclear exchange model. The next chapter will conclude the report with some observations concerning BRIK's behavior and will suggest areas for further study.

VII Conclusion

BRIK is unlike any other nuclear exchange model because it uses preemptive goal programming (G.P.) to meet all of the allocation rules and to assign weapons to targets. Because of G.P., BRIK is extremely versatile. This final chapter is divided into three sections which discuss this versatility. First, a summary of the modeling process will be presented in order to provide a perspective of the underlying process which led to the development of BRIK. The second section will provide a discussion of some potential applications of BRIK. Finally, the third section will present observations concerning some of BRIK's quirks.

Summary

This project started as a continuation of a thesis defended in December 1982 (Ref 27). Michael C. Wambsganss demonstrated that auxiliary variables in linear programming constraints could be used to allocate weapons to targets in order to meet user-defined damage expectancy goals. Wambsganss employed a damage function approximation where his model allocated non-integer weapons to each target in a target class. Next, to increase solvability, he linearized the damage function with logarithms, permitting the use of LP. Finally, he included auxiliary variables in each target constraint. With these auxiliary variables appearing in a single objective function, Wambsganss demonstrated that an allocation could be made to meet user-defined DE goals. From

this preliminary work, BRIK's evolution into an elaborate, user-friendly model is a noteworthy advancement.

Flexibility was the first requirement placed on BRIK. A model is flexible if it can handle many different types of allocations. In a nuclear exchange model, this is best handled by permitting numerous allocation rules. Various models were reviewed to determine what rules and options are currently in use. From the list compiled by this review, BRIK's hedging options were developed. The model also includes the ability to prevent a designated weapon class from attacking any specific target class. These options give BRIK the flexibility that was originally desired.

The next requirement placed on BRIK was the ability to compute single shot probability of survival (SSPS). Wambsganss' model computed SSPS given the vulnerability of a target class expressed as peak overpressure--a PSI number. Because this system is inaccurate, the authors decided that the usefulness of BRIK would be increased if the model included the nuclear targeting vulnerability (VN) system used by the Defense Intelligence Agency. This was accomplished by using Subroutine DPDX from the SAC BLUE SWADE model. Because VN numbers may not always be available, BRIK also retained the ability to compute SSPS from vulnerability expressed as PSI numbers.

It was also felt that BRIK should be user-friendly. Large, complicated models sometimes require many weeks of

study before they can be adequately used. To reduce this learning time for BRIK, the program was written to supply prompts which guide a user through each section of a problem. BRIK then uses the provided information to perform all of the mathematics internally. For example, to build a hedge which enforces a minimum amount of damage on a designated target class resulting from a particular set of weapon classes, a user only has to perform a few simple steps. First, the user is asked to choose the desired type of hedge from a menu. Next, he/she is prompted to supply the number of the designated target class and the minimum DE. Finally, the user is asked to input the weapon class numbers that should be used to attain that DE. BRIK now uses all of this information to internally build the necessary constraint.

Another requirement imposed on BRIK was transportability. Since use of a local library package might inhibit the usefulness of this model, the literature once again was searched for an appropriate allocation routine. At first, the search was for linear programming packages, but, as the design of BRIK progressed, goal programming routines were also sought. PAGP, the Partitioning Algorithm for Goal Programming, was chosen because it offered several efficiencies of which BRIK could take advantage. These included partitioning of the constraints, an indicator array which reduced the number of variables that needed to be considered for entry into the basis, and special stopping rules. The routine is copy-

righted, therefore it cannot be used for commercial gain. However, the Association for Computing Machinery gave permission to include it in this model.

The use of G.P. permitted BRIK to evolve into a multipurpose model. BRIK has the capability to build different objective functions allowing vastly different questions to be asked about the same LP matrix. This capability will be further discussed in the next section.

Use

Models are nothing but "black boxes" (Ref 24:53). One should always remember that BRIK is only a model and behaves according to the algorithms written into it. Given the same input, it will give the same output; but two different analysts may come up with different answers to the same question. This is due to the fact that allocations are made within a set of constraints which should reflect strategy and operational considerations. These constraints can be input as allocation rules. Because of BRIK's flexibility, a virtually endless variety of allocation rules are available. Also, because of the different objective functions, various kinds of analyses can be performed.

BRIK can be used to analyze build-down strategies. In the spring of 1983, US Senator William Cohen (D-Maine) suggested a strategy to stabilize the current nuclear forces. He presented the idea of removing a number of weapons from the existing arsenal in exchange for new weapons added to the

force. BRIK is a tool that can aid analysis of various build-down strategies. The following is a description of how the model could be used for this purpose. It is first necessary to determine the DE capability of the existing nuclear arsenal. Next, a problem can be run using the current forces, a current target base, and the DE goals that the force can attain. Then, the sensitivity section of BRIK is used to increase weapon availability on some select system. If the first objective function--meet goals with some minimum aspect of the available arsenal--is used, some weapons will go unused when the problem is rerun. If the unused weapons are different than the weapons added, the unused weapons could become candidates for build-down.

BRIK could also be used to determine future force structure. The second objective function can be used to "build" the number of weapons needed to meet user-defined DE goals. To choose between alternative systems, the following steps could be performed. A weapon base that includes the alternative weapon classes should be entered along with a target base and appropriate DE goals. After inputting all of the appropriate allocation rules, the problem would be run. The output would show which systems were "built," thus giving the analyst indication as to which system to purchase.

A third type of analysis that BRIK is well suited to perform is a sensitivity analysis. Because of the interactive qualities of this model, an analyst can sit at a termi-

swers to these questions are displayed on the screen enabling the analyst to immediately see the results of the changes made to the problem. This sensitivity capability is a step toward building the tools of strategic force analysis around the machines used, making the machines extensions of human thought.

BRIK has a tremendous amount of potential to aid analysis in many different areas of nuclear exchange as well as a versatility heretofore unknown. However, it is not perfect. The following section describes two areas a user may find as shortfalls to the model.

Observations

There are two aspects of BRIK's behavior which may concern potential users of this model. The first area deals with BRIK's current lack of capability to maximize damage expectancy while the second area concerns the damage function approximation. After presenting observations depicting BRIK's behavior in these two areas, a discussion of possible corrections follow.

The first area of concern is BRIK's lack of capability to maximize DE. Given a weapon base and a target base, most nuclear exchange models will maximize damage expectancy (Ref 24:53). To see if BRIK has this capability, the test problem used to compare BRIK with the Arsenal Exchange Model (AEM) was run with DE goals of .98 for each target class. The results are in Table X. Comparing this table with the

solution from AEM in Table VII, it can be seen that BRIK's allocation is not close. Also, the total value destroyed by BRIK, 1367.2, is considerably lower than the damage achievement by AEM of 1770. Therefore, in its current form, BRIK does not maximize damage expectancy.

The second area of concern is the "optimistic" DE from the damage function approximation described in Chapter III. Figure 18 shows the results of a small problem where an achievement of .5 was attained on target class "civil." From the figure it can be seen that 4.08 weapons were assigned to the class. Since the class originally had 10 targets and there are 5 targets remaining, 5 targets were destroyed by 4.08 weapons! These areas are mentioned because they may be of concern to potential users of BRIK in its current form. However, this does not imply that both limitations need be permanent.

It is believed that BRIK's current lack of capability to maximize DE can be solved. One potential technique is an iterative scheme that systematically changes DE goals on the target constraints. The iterations should concentrate on allocating weapons to targets that have not been assigned any weapons before allocating weapons to targets that already have weapons asigned. This could possibly solve the first area of concern. Giving BRIK the capability to maximize DE seems realistic and would be a good addition to the model.

TABLE X

Test Problem BRIK Solution: High DE Goals

****

TOTAL VALUE DESTROYED WAS 1367.2.

TGTNAM	WPNCLASS	NUMBER ASSIGNED	GOAL	ACHIEVEMENT	NUMBER REMAINING
civil	MMII	256.45	.98	.98	2.8
	FB111	49.66			
local	Poseid	456.39	.98	.98	4.3
c3i			.98	.00	450.0
icbm			.98	.00	1000.0
lcc			.98	.00	200.0
subpts	FB111	48.58	.98	.98	.4
irbm	Poseid	104.84	.98	.98	3.0
	B52grv	208.18			
afbase	B52grv	215.82	.98	.98	2.0
stores	MMII	193.55	.98	.57	184.9
facil	MMIII1	750.00	.98	.98	10.4
	Poseid	478.77			
	FB111	261.75			

WPNAME	NUMBER	USED	REMAIN
MMII	450.0	450.0	.0
MMIII1	750.0	750.0	.0
Poseid	1040.0	1040.0	.0
B52grv	424.0	424.0	.0
FB111	360.0	360.0	.0

TOTAL VALUE DESTROYED WAS 5.0.

TGTNAM	WPNCLASS	NUMBER ASSIGNED	GOAL	ACHIEVEMENT	NUMBER REMAINING
civil	MMII	4.02	.95	.50	5.0
local			.95	.00	10.0

WPNAME	NUMBER	USED	REMAIN
titan	49.0	.0	49.0
MMII	45.0	4.1	40.9

Figure 18. Sample Problem

However, this deficiency does not negate the advantages gained from the availability of this tool. It matches the methodology of sequential achievement of damage goals, as outlined in Reference 24.

The damage function approximation in BRIK results from assigning non-integer weapons to each target in a class. It is used internally in two different places. First, it is used to build all of the target constraints and the hedging constraints which deal with minimum or intermediate DE goals. Second, the damage function approximation is used in the output section to compute the achievement on each target class. To eliminate the damage function approximation, the model could be completely reformulated where the decision variable is some number of targets that receive integer

weapons. From the authors' points of view, this would be an extremely difficult task. It is possible, though, to take the allocation and recompute the damage achievement. For example, in Figure 18, since 4.08 weapons were assigned to target class "civil," the achievement could be computed by assuming 4.08 targets were each allocated one weapon and 5.92 targets were allocated no weapons. Recomputing the DE in this manner would allow for a more accurate representation of the achievement, however, it would also result in some inconsistency in the model. Again, using Figure 18 as an example, 4.08 MMII warheads were assigned to target class "civil" by using a hedge that asked for a DE goal of .5. The current output reflects that this goal was met and there were remaining weapons. If the achievement was recomputed as described above where 4.08 targets received 1 weapon and 5.92 targets received no weapons, the achievement would have been reported as .33, compared with the desired goal of .5. By making conparisons between BRIK's non-integer allocations with integer solutions, this difference in achievement is the largest one that was found. While the current damage function approximation makes weapons appear optimistic, the difference is consistent within the model as it is currently formulated. Also, reported DE is optimum if fractions of weapons could in fact be allocated. Therefore, BRIK's use as an analysis tool is not compromised by the presence of this damage function approximation.

Conclusion

BRIK started with an LP formulation and a single objective function. It evolved into a model that contained a unique application of G.P., from which a tremendous amount of capability is enjoyed. The deviational variables and how they are handled in the different objective functions is the key to BRIK's flexibility and potential for multipurpose application. BRIK permits the user to determine its application and to examine numerous objectives and strategies providing a virtually "custom designed" model tailored to the individual user's requirements.

Bibliography

1. Allot Input Requirements. Hq SAC:XPXF, Force Structures Division, Omaha, Nebraska. Author and date unknown.

2. Arthur, Jeffrey L., and A. Ravindran. "PAGP, A Partitioning Algorithm for (Linear) Goal Programming Problems." ACM Transactions on Mathematical Software, 6 (3): 378-386 (September 1980).

3. Battilega, J. A., and W. L. Cotsworth. Multiple Criteria, Strategic Force Analysis, and the Arsenal Exchange Model. Science Applications Inc., Englewood, California. March 1977. (SAI-77-058-DEN).

4. Bennett, Bruce W. Assessing the Capabilities of Strategic Nuclear Forces: The Limits of Current Methods. Rand, Santa Monica, California. June 1980. (AD A111 725).

5. Benninger, Gilbert C., Paul J. Castleberry, and Patsy M. Mcgrady. Mathematical Background and Programming Aids for The Physical Vulnerability System For Nuclear Weapons. Defense Intelligence Agency, Washington D. C. November 1974. (AD B010 375L).

6. BLUE SWADE Listing. Hq Sac:XPXF, Force Structures Division, Omaha, Nebraska.

7. Bracken, Jerome, and James T. McGill. "A Convex Programming Model for Optimizing SLBM Attack of Bomber Bases," Operations Research, 21: 30-36 (1973).

8. Brewer, Wallace D. The Fundamentals of Weapons Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, March 1983. (CN F33601-81-C0185).

9. David, K. H., and F. Lemus. "An Optimum Allocation of Different Weapons to a Target Complex," Operations Research, 11: 787-794 (1963).

10. Davis, Lynn Ethridge, and Warner R. Schelling. "All You Ever Wanted To Know About MIRV and ICBM Calculations But Were Not Cleared To Ask." The Journal of Conflict Resolution, 17 (2): 207-214 (June 1973).

11. Day, Richard H. "Allocating Weapons to Target Complexes by Means of Nonlinear Programming," Operations Research, 14: 992-1013 (1966).

12. Denesia, T. E. A Framework for the Comparison of Four Military Strategic Simulation Models. MS Thesis. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, December 1978.

13. Deterrence and Survival in The Nuclear Age (The "Gaither Report" of 1957). US Government Printing Office, Washington D.C. 1976.

14. Glasstone, Samuel, and Philip J. Dolan. The Effects of Nuclear Weapons. Third Edition. U.S. Department of Defense, U.S. Government Printing Office, Washington D.C. 1977.

15. Grotte, Jeffrey H. "An Optimizing Nuclear Exchange Model for the Analysis of Nuclear War and Deterrence," Operations Research, 30:428-445 (1982).

16. Hillier, Fredrick S., and Gerald J. Lieberman. Introduction to Operations Research. Third Edition. Holden-Day, Inc. San Francisco, California. 1980.

17. Hodson, William T., William G. Goodyear, and Wolhurt B. Goethert. A Linear Programming Approach to Weapon Allocation. Research Report 71-1, U.S. Air Force Academy, Colorado Springs, Colorado, March 1971.

18. Ignizio, James P. Linear Programming in Single- and Multiple- Objective Systems. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1982.

19. Manne, Alan S. "A Target-Assignment Problem," Operations Research, 6: 346-351 (1958).

20. Martin Marietta Corporation. Mathematical Formulation of the Arsenal Exchnge Model. Denver, Colorado, June 1973.

21. Miercort, Frederic A. "Optimal Allocation of Missiles Against Area and Point Defenses," Operations Research, 19: 605-617 (1971).

22. Perkins, F. M. "Optimum Weapon Deployment for Nuclear Attack," Operations Research, 9: 77-94 (1961).

23. Ross, James A. The Allot Computer Program. Institute for Defense Analysis, Systems Evaluation Division, Washington, D.C. April 1979. (AD A069 111).

24. Seiler, George J. Strategic Nuclear Force Requirements and Issues. Air University Press, Maxwell Air Force Base, Alabama, February 1983. (RN AU-ARI-82-1).

25. Soland, Richard M. "Optimal Defensive Missile Allocation: A Discrete Min-Max Problem," Operations Research, 21: 590-596 (1973).

26. Specht, R. D. "The Nature of Models." Systems Analysis and Policy Planning: Applications In Defense. Elsevier Scientific Publishing Company, Amsterdam. 1977.

27. Wambsganss, Michael C. A Linear Programming, Multiple Objective Approach to Nuclear Exchange Modeling. MS Thesis. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, December 1982.

28. Zelany, Milan. "Multiple Objectives in Mathematical Programming: Letting the Man In," Computers and Operations Research, 7: 1-4 (1980).

APPENDIX A

User Guide

This guide will familiarize the user with many of the features of BRIK. It will step through BRIK in the order of appearance of the various model functions. Examples will be supplied when necessary for clarity.

BRIK was developed to be "user friendly". The standard of "user friendly" is taken to mean that anyone with the requisite data and limited background in either computer or mathematical programming could use the model for an allocation. Since BRIK displays prompts and menus on the screen to guide the user through the model's functions, the user needs access to a computer equipped with a monitor.

Many of the prompts which appear on the screen are questions that require answers. Realizing that a user could input an illogical response or an incorrect piece of data, the model incorporates many internal checks to guard against the possibility of an unintentional mistake. An example of a check can be found at any yes or no (y/n) question. If the user gives an illogical response, the cursor drops a line and waits for a correct response, y or n. Also, the model discourages the user from inputting inappropriate data. For example, probabilities are accepted only if they are between 0 and 1. The procedure is the same as the yes/no question; the cursor will wait for the proper input.

On some menus, the user must enter a 0 to terminate that

section of the model. It has been discovered that the VAX computer interprets any non-numeric character as a 0 when it is expecting an integer. It is unknown if any other computer has the same logic. Therefore, the user should be cautioned that if any character value is entered for an integer, that particular model section can terminate prematurely.

BRIK can be used for two types of analysis. The first type is used for any new problem where the user must input data, select the type of objective function, and enter the allocation rules for the problem. The second type enables the user to conduct a sensitivity analysis on some of the input parameters. The sensitivity section assumes that an initial type 1 analysis has been accomplished.

This user guide will be broken up into two main sections. The first section will discuss the interaction required to conduct a new analysis. The second section will describe the capability of the sensitivity section.

NEW ANALYSIS

This section will describe the four major sub-areas of a new analysis: input requirements, selection of an objective function, allocation constraining rules, and BRIK's output data.

Input Requirements. BRIK allows the user to input data interactively or through user-designated files. The data input is divided into three subsections: target data, weapon data, and damage expectancy (DE) goals. This section will

describe the type of data needed by BRIK and the types of operations and editing that are available.

Data is entered in the following order: target base, weapon base, and DE data. Since BRIK follows the same basic operation of input for all three data types, the following explanation applies to all three types. The first decision the user must make is whether the data is to be entered interactively or from a file. If there has not been any previous analysis which created a usable data base stored in a user-designated file, the data must be entered interactively. If the data is entered interactively, BRIK will guide the user with a series of questions. These questions insure that the required data is input in the correct fashion.

Some input data is entered as a group and there is no individual check for data correctness. Therefore, at the end of all the data input sections, any illogical or incorrect data will be converted into a form that BRIK can use. These default values, except for probabilities and percentages, will be indicated in the input data lists. Probabilities and percentages are not allowed to be greater than 1 or less than 0. If data is input greater than 1, BRIK will decrease the value to 1; and if a number less than 0 is input, it is corrected to 0. It is also possible to enter some data that is physically too large to display on the screen due to format limitations. Even though the data will be read correctly, all screen output for this data will be a series of stars. These

format limitations will be listed in the data lists.

It is possible to build a data file independently of BRIK, but it is easier to input the data interactively and allow the model to build an external file. If the data base is input from an external file, the user must insure the file exists prior to operating BRIK. If a data file is built, its name must have less than seven characters. Since the file must be formatted, it is easier to allow BRIK to create the file.

The rest of this section describes the input requirements for the three types of data bases used in BRIK. Each section will list and briefly describe the required parameters and discuss the types of editing that can be done with BRIK.

Target Input: The following is a list of required target input data:

Class name	--	Maximum of 6 characters
Number in the class	--	Integer, format limit = 9999
Target vulnerability	--	VNTK number or peak overpressure needed to achieve a desired level of damage. If the VNTK number is incorrectly input, the program will terminate after the DE goals are input.
Target diameter	--	The diameter of a circle in nautical miles (NM) that encompasses 95% of the target. If the target is a point target the diameter input is 0. The default is 0 if a negative number is input.
Target category	--	FORCE = f VALUE = v, default value MILITARY = m

Target class priority -- Integer 1 to 7, default is 1 if a value less than 1 is input and 7 if a value greater than 7 is input.

Value -- The worth of a target to the attacker. The default is 1.

The program also has an algorithm to calculate values for force type targets. If the user exercises this option, the following additional target data is needed:

Target reliability -- Probability that the weapon located at force target will be successful against the attacker.

CEP -- Nautical Miles (NM), defaults to .00001

Yield per warhead -- Megaton Yield, format limit is 999.99

Warheads per target -- The number of warheads located at each target. If the target is a submarine base, a possibility of many warheads exists. Format limit is 99

If the force value algorithm has been selected, the user must enter values for any value or military target classes so that the relative values between the classes is represented.

Target Editor: The first item the user sees after inputting the target data is a menu for editing the data (Figure A-1). This menu allows the user to either view or change any target input parameter and to add, replace, or delete an entire target class. Again, the editor sections ask self-explanatory questions. It should be noted that after making any change, the user should view the data before continuing in the program. This will insure the new data set is correct.

If data is to be input via an external file, the user can manually edit the data file before running BRIK (Figure A-2). The user should be cautioned not to displace any data by even one space, as the format BRIK uses to read the data is an exact one. Also, if the user adds or deletes a class, the first number in the file must be changed. This number designates how many classes are in the file.

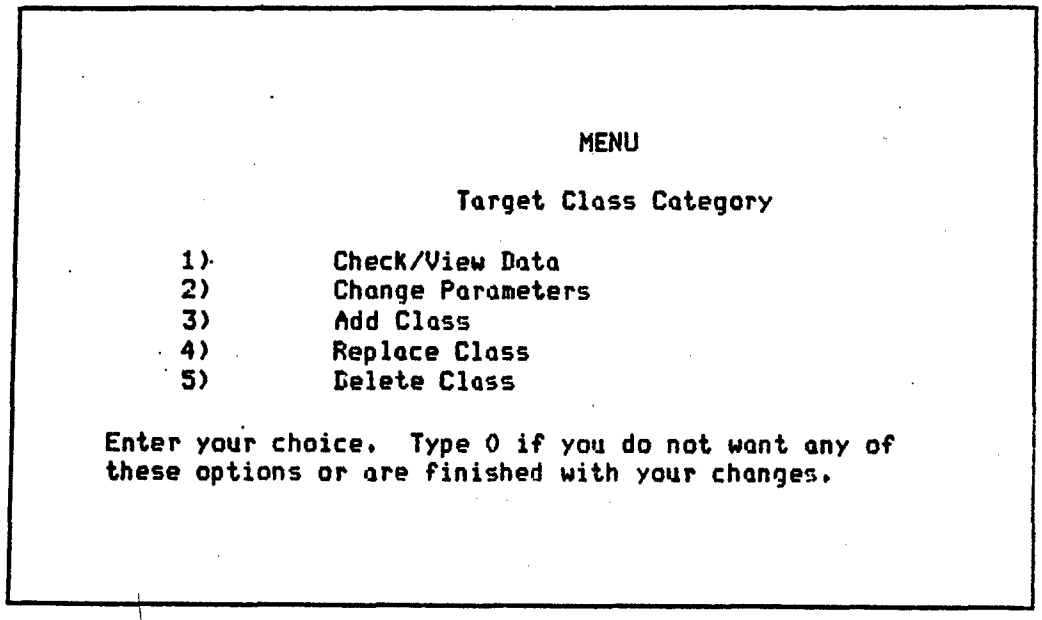


FIGURE A-1. Target Editor Menu

```
 5
 NAME   NUMB    VUL    DIA TYPE PRIO VAL   REL  CEP  YLD  WHDS
ldrshp   200 20q5    .20    v    1   1.00  .00  .00  .00    0
icbm     500 45p0    .00    f    2   1.00  .00  .00  .00    0
slbm      30 30q0    .30    f    2   1.00  .00  .00  .00    0
pol      500 30     1.00    v    3   1.00  .00  .00  .00    0
afbase    50 35     1.00    m    3   1.00  .00  .00  .00    0
>
```

FIGURE A-2. Example of Target Base External File

Weapon Input: The following list contains required weapon input parameters. As in the target input section, data can be input interactively or by a user-designated file.

Class name -- Maximum of 6 characters

Number in the class -- Integer, format limit is 99999

Warheads per weapon -- The amount of warheads per weapon, carrier, format limit is 999

Reliability -- Probability that the weapon will be successful.

CEP -- Nautical Miles (NM), defaults to .00001

Weapon yield -- Megaton Yield, format limit is 999.99

Daily alert rate -- Percentage of the weapon class useable for the allocation

Generated alert rate -- Percentage of the weapon class useable for the allocation

The daily and generated alert rate data can be used in a counterstrike scenario where the daily alert rate can represent the pessimistic percentage of surviving weapons and the generated alert rate can represent the optimistic percentage of surviving weapons.

Weapon Editor: The weapon editor is similar to the target editor in the types of available options (Figure A-3). It also has self-explanatory questions to help the user through the section. The weapon data file is similar in construction to the target data file (Figure A-4). It can be manually edited, but again caution is advised.

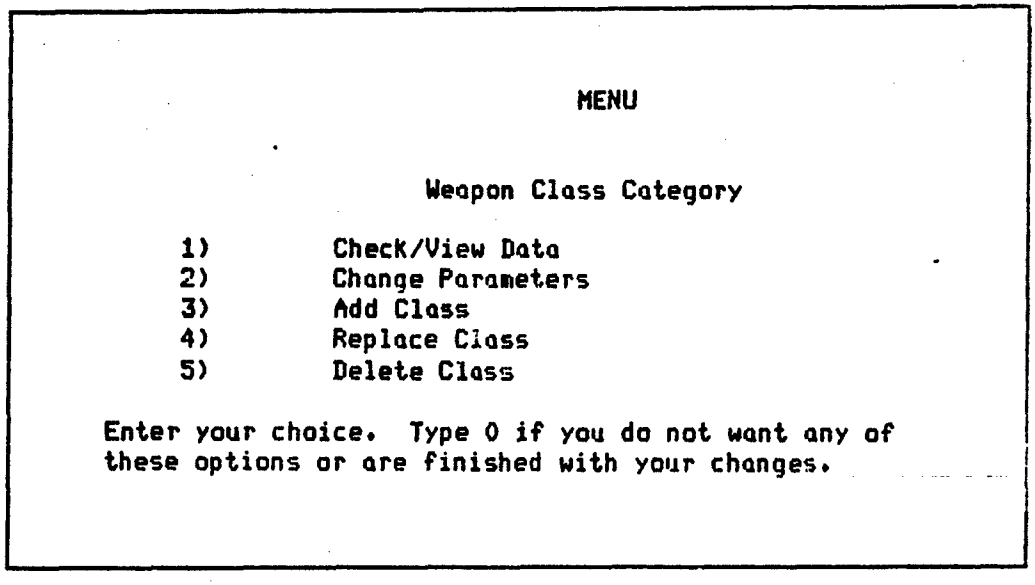
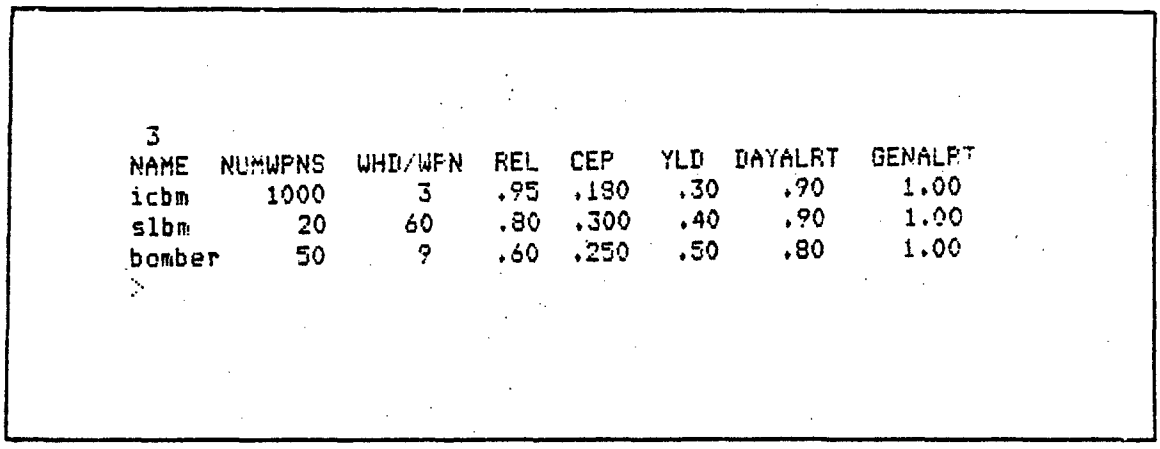


FIGURE A-3. Weapon Class Editor



```
 3
NAME    NUMWPNS  WHD/WPN  REL   CEP    YLD   DAYALRT  GENALRT
icbm       1000        3  .95   .180   .30     .90      1.00
slbm         20       60  .80   .300   .40     .90      1.00
bomber       50        9  .60   .250   .50     .80      1.00
>
```

Figure A-4. Example of Weapon Base External File

Damage Expectancy (DE): Damage Expectancy is the percentage of destruction the user wants to inflict on a particular target class. DE can be input from a file or interactively. If there have been any target class additions or deletions in the target editor, the model will only accept DE input interactively. The user can enter DEs by either target category (m, v, or f) or by individual class (Figure A-5). If the DEs are entered by category, all the target classes which are classified in that category will have the same DE. If a DE value greater than or equal to 1 or less than 0 is input from a file, BRIK will either reduce it to .999 or raise it to 0, respectively. After DE values have been entered, the user can check the DE values and make corrections. Again, BRIK guides the user with questions and prompts.

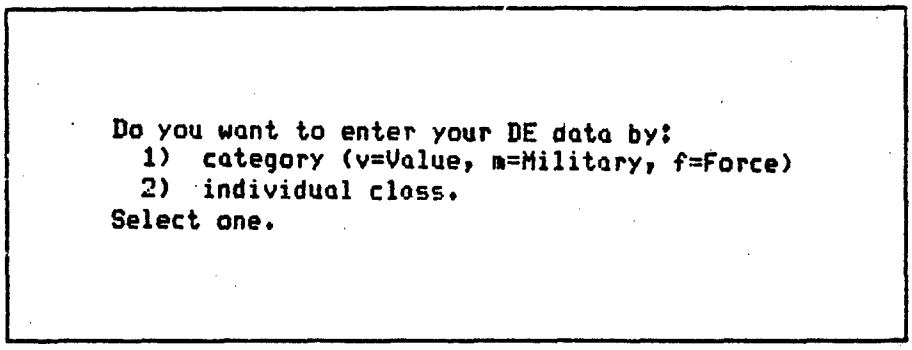


Figure A-5. Example of DE Menu

Once all the DE values have been input and edited, BRIK will compute the single shot probabilities of survival (SSPS). This is indicated by a "PLEASE STANDBY" on the monitor screen. The single shot probabilities of survival are not listed on the screen and cannot be externally changed. They are listed in the file called "SPARSE".

When the calculations are complete, BRIK will ask the user what type of alert rate his weapon base is on. This will determine what percentage of the target base to use for the allocation.

This concludes the entry of all required parameters. The weapon availability and target characteristics along with individual DE goals have been specified. The next two sections describe the objective functions and constraining rules that can be used in BRIK.

Objective Functions. There are three objective functions available in BRIK (Figure A-6). The first objective function attempts to meet all of the user's goals with the available arsenal. If all goals can be met, the model will hold the remaining weapons in reserve. The arsenal characteristics used in the allocation will depend on the "extreme goal". Extreme goal options will be discussed in the Constraining Rules section.

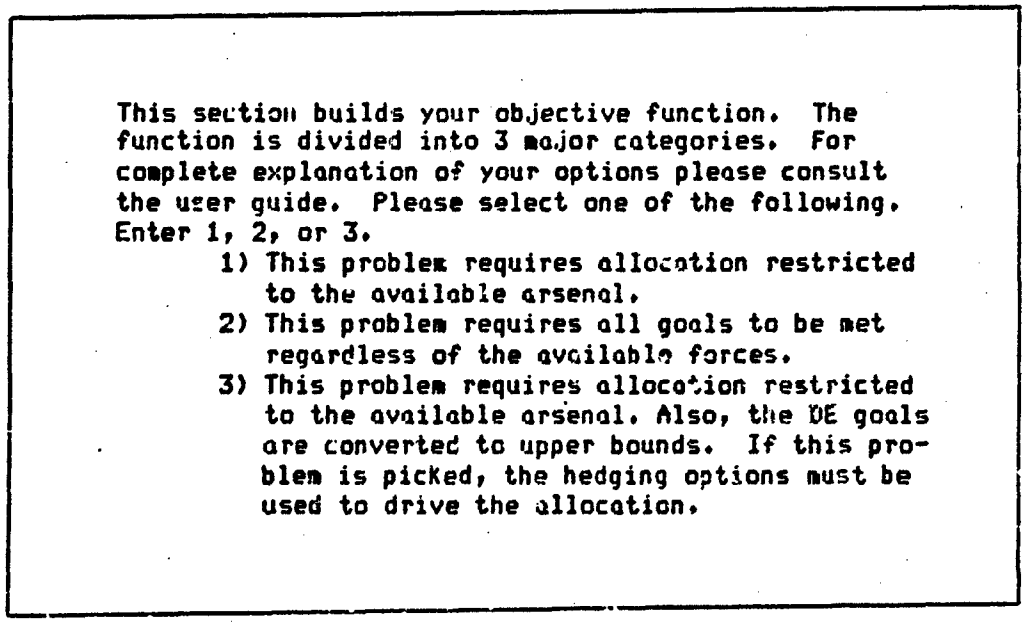


FIGURE A-6. Objective Function Menu

The second objective function forces all DE goals to be met regardless of the weapon availability. It can be used to determine the type and number of weapons to add to an existing arsenal, or to create a new arsenal to meet DE goals. The arsenal created or expanded will only consist of those weapon

classes input in the weapon class input section.

The third objective function bounds the problem with both weapon availability and DE goals. Since the DE goals are not used to drive the allocation, hedges must be used (see next section). The Target DEs will no longer be treated as goals. They set the upper limit on the amount of destruction to any target class. The only way an allocation can be run using this objective function is to add hedging constraints to the problem or by using the fifth extreme goal and allocating the entire arsenal. If no hedges are added or one of the other extreme goals is used, BRIK will not perform any weapon allocation.

Constraining Rules. The following options allow the user to select a variety of rules to constrain the allocation. There are three types of rules available in BRIK: an extreme goal, inappropriate weapon/target assignments, and several types of hedging.

Extreme Goal: This is the lowest priority goal in each objective function. Its purpose is to drive the allocation to a unique solution. Sometimes, however, the rule will have no effect. For example, using the first objective function, if all the weapons were used in the allocation, this rule will be ignored. BRIK has five different extreme goals (Figure A-7). The first four goals can be used with any of the three objective functions, while the fifth goal is only available with the third objective function.

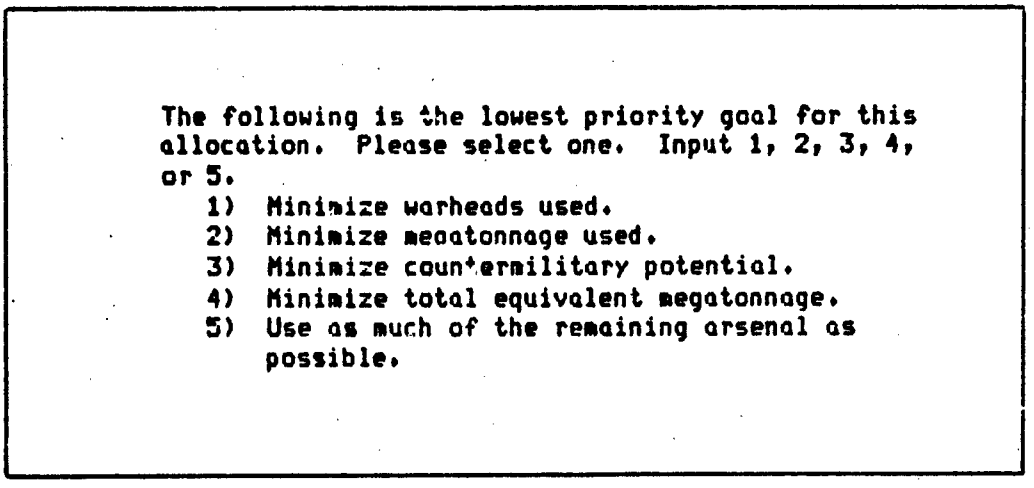


Figure A-7. List of BRIK Extreme Goals

Inappropriate Weapon/Target Interaction: After the extreme goal has been selected, BRIK will ask for any inappropriate weapon/target interactions. This function can be used if there are some weapon classes that should not attack certain target classes, either for military or political reasons. An example of a time urgent target is a bomber base on alert. It may be inappropriate for a weapon system which will not arrive until all the bombers have launched to be allocated against it. Range limitation is another reason to use this allocation rule. For example, suppose that a target class had some targets out of range of a specific weapon class. The target class could be divided into two subsets. One set would incorporate all of the targets that could not be reached by the weapon class, and the rule to

prohibit the weapon class from attacking the subset would be added to the problem.

Hedges: Hedges are additional constraints which are added to further control the allocation. There are seven types of hedges available (Figure A-8). After a hedge is created, the model will always return to the menu. Also, once created, the hedge cannot be deleted from the problem, so the user should be cautious.

The first two hedge types are similar, both enforce minimum acceptable levels of damage on certain target classes. However, they differ in the weapon set used to meet the constraint. Hedge 1 insures the minimum DE level is accomplished using any of the available arsenal, while hedge 2 attempts to meet the minimum DE goal with some user-designated subset of weapons. These minimum DE goals should not be confused with the DE goals input in the DE data section. Since hedges appear at a higher priority in the first two objective functions, the minimum goals will all be met (or attempt to be met) prior to any allocation to satisfy the DE goals. Also, since objective function 3 only uses the DE goals to provide an upper DE limit for all target classes, these hedges are needed to help drive the allocation.

```
This section permits selection of user defined goals.
There are seven types to choose from.  Select any
particular type by inputting the appropriate number.
For further information about hedging, see the users
guide.
You have space to select 20 hedging constraints.
The following is a list of hedging types available.
Type 0 to exit this section.
  1) Enforce a minimum level of damage on a particu-
     lar target class.
  2) Enforce a minimum level of damage on a particu-
     lar target class using a specific set of weapons.
  3) Enforce an upper level of damage on a particu-
     lar target class resulting from a specific set
     of weapons.
  4) Restrict the number of a class of weapons which
     can be allocated to a target class.
  5) Build your own constraint.
  6) Enforce a minimum level of damage on a particu-
     lar set of target classes.
  7) Restrict the number of weapons which can be al-
     located to each target in a particular target
     class.
```

Figure A-8. BRIK Hedging Options

The third hedge is designed to limit the damage on a particular target class resulting from a specific set of weapons. For example, if it is desired that an ICBM target class receive at most fifty percent of its DE from bomber and SLBM weapon classes, this hedge should be used.

The fourth hedge restricts the number of weapons from a particular weapon class that can be allocated to a designated target class. By restricting the amount of weapons from a designated weapon class that can be allocated to a specific target class, BRIK is forced to use other weapons to attain the target class DE goal. This allows the remaining weapons

of the designated class to attack other targets.

The fifth hedge allows the user to build his own constraint. This hedge should be used only if there is a good working knowledge of how BRIK builds constraints. This hedge allows the user to create hedges that are not indigenous to BRIK, such as a common carrier constraint. If two weapons are carried aboard a common carrier in a specified ratio, the weapons must be used in that ratio. For example, if equal numbers of two weapon classes had to be carried on a bomber, the model should allocate exactly the same amount of each weapon type in the problem. The following constraint forces an equal amount of each weapon class to be used to attack two target classes:

$$X_{11} + X_{12} - X_{21} - X_{22} = 0$$

The first two terms, X_{11} and X_{12}, indicate the amount of the first weapon class allocated to target classes 1 and 2. The second two terms, X_{21} and X_{22}, indicate the amount of the second weapon class allocated to target class 1 and 2. If both weapons are used equally, the difference between the number of each used must be 0. If the bomber could carry 2 type 2 weapons for each type 1 weapon, the constraint will change to:

$$X_{11} + X_{12} - 1/2X_{21} - 1/2X_{22} = 0$$

This forces one type 1 weapon to be used for every two type 2 weapons used.

The sixth hedge sets a minimum level of acceptable damage against a set of target classes. This constraint should be used carefully. For example, if the user wanted at least 30% destruction against a set of three target classes and one of these classes was extremely easy to kill, BRIK could attack the easy-to-kill target class and leave the other two classes untouched. It is suggested that this hedge be used only with the third objective function because all of the target classes have upper bounds on the amount of damage inflicted on them.

The seventh hedge restricts the number of weapons that can be allocated against each target in a target class. This constraint could be used to force an upper limit of total weapons that can be used against an individual target class. Also, if one of these constraints were added for each target class a targeting strategy of allowing at most one weapon per target for every target in the entire base could be enforced. This hedge could be used to force an integer solution if the DE goals are set extremely high and there are enough weapons in the weapon base to meet this hedge.

After the final hedge is input, "PLEASE STANDBY" will appear on the monitor screen. This indicates that BRIK is creating the matrices and files necessary to run the problem. Once the files are built, BRIK will begin the allocation algorithm. This is indicated by prompts which indicate the priority currently being worked. The allocation function can take anywhere from a few seconds to several tens of minutes,

depending on problem size and the time sharing system used on the computer.

Output. The purpose of this section is to explain the output. The following example uses the target and weapon data listed in Figures A-2 and A-4 with a type one objective function.

BRIK displays the output data on the monitor screen (Figure A-9) and also places the output into an external file called "alloc" (Figure A-10). The screen output gives the user three types of information: the total value of all the targets destroyed, the percentage of each target class destroyed including a detailed list of the type and number of each weapon class allocated against each target class, and the

TOTAL VALUE DESTROYED WAS 934.0.

TGTNAM	WPNCLASS	NUMBER ASSIGNED	GOAL	ACHIEVEMENT	NUMBER REMAINING
ldrshp	icbm	47.60	.90	.90	20.0
	slbm	222.63			
icbm	icbm	2905.98	.80	.80	100.0
slbm	icbm	31.06	.80	.80	6.0
pol	slbm	303.36	.60	.60	200.0
afbase	icbm	15.36	.60	.60	20.0

WPNAME	NUMBER	USED	REMAIN
icbm	3000.0	3000.0	.0
slbm	1200.0	526.0	674.0
bomber	450.0	.0	450.0

Figure A-9. Screen Output

number of weapons of each weapon class used in the allocation. "Alloc" contains two sets of information. The first set contains data from the goal programming package and the second set is a duplication of the screen output. Set one consists of a five-column table. The first column lists the subscript number of the priority or of the variable in that row. The second column contains the objective value for each priority. The third column contains the values of the decision variables, and the fourth and fifth columns list the values for the positive and negative deviational variables for each goal respectively (Figure A-10). The length of the table is a function of the greater number of either decision variables or goals in the problem.

The first table in "Alloc" contains a lot of information. For example, when a problem uses a type 1 or a type 2 objective function, if the number in column 2 for row 2 is non-zero, then at least one of the hedging rules could not be met. In the "X OPT" column, the specific weapon/target assignments can be found. To compute the location of the weapon/target assignment, use the following formula

$$(j + (i-1) \times \text{number of target classes})$$

where j is the target class and i is the weapon class. For

RUN NUMBER 1.

THE OPTIMIZATION ENDED ON SUBPROBLEM 6
THERE WERE 9 CONSTRAINTS IN THE FINAL OPTIMAL TABLEAU.

****

O OUTPUT SUMMARY

OSUBSCRIPT	A OPT	X OPT	POS DEV	NEG DEV
1	.0000	47.5996	.0000	.0000
2	.0000	2905.9756	.0000	.0000
3	.0000	31.0639	.0000	.0000
4	.0000	.0000	.0000	.0000
5	.0000	15.3609	.0000	.0000
6	3525.9973	222.6349	.0000	.0000
7		.0000	.0000	674.0029
8		.0000	.0000	450.0000
9		303.3623	3525.9973	.0000
10		.0000	.0000	.0000
11		.0000	.0000	.0000
12		.0000	.0000	.0000
13		.0000	.0000	.0000
14		.0000	.0000	.0000
15		.0000	.0000	.0000

****

TOTAL VALUE DESTROYED WAS 934.0.

TGTNAM	WPNCLASS	NUMBER ASSIGNED	GOAL	ACHIEVEMENT	NUMBER REMAINING
ldrshp	icbm	47.60	.90	.90	20.0
	slbm	222.63			
icbm	icbm	2905.98	.80	.80	100.0
slbm	icbm	31.06	.80	.80	6.0
pol	slbm	303.36	.60	.60	200.0
afbase	icbm	15.36	.60	.60	20.0

WPNAME	NUMBER	USED	REMAIN
icbm	3000.0	3000.0	.0
slbm	1200.0	526.0	674.0
bomber	450.0	.0	450.0

Figure A-10. Output File "Alloc"

example, in Figure 9, there are five target classes and three weapon classes in the problem. To find out how many weapons of class 2 (slbm) were assigned to target class 4 (pol), the formula yields

$$(4 + (2-1) \times 5) = 9$$

Looking in column "X OPT" at row nine, yields 303.36, which agrees with the screen output. The deviation from each constraint's right hand side is listed in the "POS DEV" and "NEG DEV" columns. The order of appearance for the constraints in the goal programming problem is as follows: the target DE goals, weapon availability goals, extreme goal, and the hedging options. Since there are five target classes used in the sample output in Figure 9, the rows which contain the deviations for the weapons availability are 6 through 8. The amount of unused type 2 weapons is 674.0, which agrees with the screen output.

Although all of the information is readily available and easier to interpret from the screen output, the goal programming package data is useful to give the advanced user a further understanding of the allocation. The ability to eliminate the first set of data from "alloc" is available in BRIK's sensitivity section, which will be discussed next.

SENSITIVITY RERUN

This section discusses the sensitivity capability of BRIK. Sensitivity consists of changing any of five types of parameters (Figure A-11). After the limitations of this

section are explained, the use of this routine will be described.The sensitivity analysis used in this section should not be confused with a linear progamming type of sensitivity. Instead of using the existing final tableau, the model reruns the problem after the parameter changes have been made.

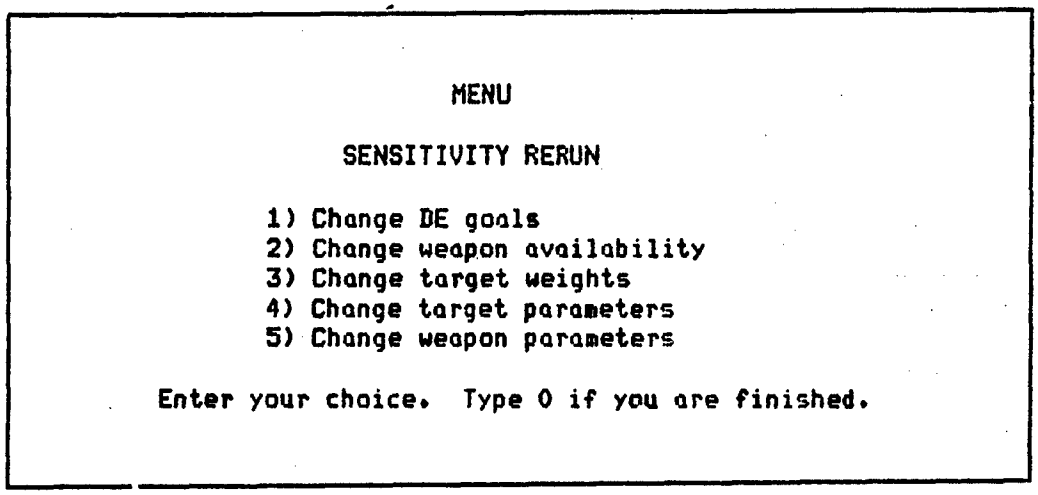


Figure A-11. Sensitivity Rerun Menu

Also, this section cannot change the objective function or the hedges used in the original problem. If the user needs to change the hedges or the objective function, a new problem must be run.

DE Goals. This subsection is used to change the DE goals of any of the target classes. It can be used to determine how

a change in DE affects the allocation.

Weapon Availability. This option can be used to see how a change in the number of weapons affects the allocation. If a problem is initially run with a weapon availability of zero, increasing the number of weapons has the effect of adding a new weapon class to the problem.

Target Class Weights. This subsection has meaning only if the first objective function was used in the original problem. If either of the other two objective functions were used, a change in weight will have no effect on the problem. The target class weight is a function of target value and target class DE. The user can simply input a new weight, or can compute a new weight based on a change in any of the three parameters. The formula is as follows:

$$\text{Weight} = \text{Target Value} \times [(1 - \text{Target DE})]$$

Target Parameters. This subsection allows the user to change two parameters, target vulnerability and target diameter. A change in either of these two parameters will affect the survivability of the target class attacked by each weapon class.

Weapon Parameters. This subsection allows the user to change three weapon parameters: weapon reliability, weapon CEP, and weapon yield. A change in any of these parameters changes the effectiveness of the particular weapon class and will affect the SSPSs for those target classes which the particular weapon class can attack.

After the user is finished making a parameter change, BRIK returns to the main sensitivity menu. When all changes are complete, the user is given the option to suppress the expanded output. The expanded output consists of all data that is entered in "Alloc" that is not displayed on the screen. Output suppression is recommended if the problem is large and it is anticipated that many runs will be made. Finally, each time the sensitivity section is entered all of the files are reinitialized to the original problem. That is, for any subsequent sensitivity run, all of the changes made during the previous runs are automatically removed before the user is given the option to make further changes.

This concludes the user guide. It was the authors' intention to provide a short summary to make BRIk useable without requiring the user to read the entire thesis. For further detail about the inner workings of BRIK, the user is referenced to the thesis.

APPENDIX B

BRIK Subroutine and Function Listing

ADDTGT -- Enables the user to add a target class to the problem

ADDWPN -- Enables the user to add a weapon class to the problem

AIJIN -- Builds the first (ntgts + nwpns) rows of the Aij matrix, using the number of targets, number of weapons, and the SPARSE matrix previously built

BOUT -- "Main" Program that calls all subroutines that are used in the allocation algorithm

CALWGT -- Calculates the value for a single target in a force target class if the user has selected this option

CHANGE -- Permits minor changes to the original problem. After the changes have been made the new problem is rerun in BRIKOUT. The following list of changes is allowed:

- Change DE goals
- Change weapon availability
- Change target weights
- Change specific weapon or target parameters

CINDX -- Computes the relative cost coefficients for each variable in the current tableau and the objective function value at the current priority

DECHEK -- Displays the damage expectancy values on the screen

DEFILE -- Reads Damage Expectancy (DE) values from an external file

DEINAC -- Allows the user to manually input desired DE values for each target class

DESAVE -- Allows the user to save to an external file his DE values

FILEIN -- After all computations are finished and the needed matrices are built, this subroutine stores the data into the appropriate files (SPARSE, AIJ, PAGPIN) for use in BOUT

FIX -- (Function) Places floating point values that are within 1.E-5 of an integer to that integer

HEADER -- Prints the name of the program on the screen when the program is turned on

HEDGE -- Allows the user to select from seven user-defined goals:

- Enforce a minimum level of target class damage
- Enforce a minimum level of damage on particular target class from a specific set of weapons
- Enforce an upper level of damage on a particular target class from a specific set of weapons
- Restrict the number of a specific class of weapons which can be allocated to a certain target class
- Build a user-defined constraint
- Enforce a minimum level of damage on a certain set of target classes
- Restrict the number of weapons which can allocated to each target in a particular target class

LISTGT -- Prints a list of target class names on the screen

LISTWP -- Prints a list of weapon class names on the screen

OBJECT -- Builds the objective function for the problem to either restrict allocation to the available arsenal or meeting all DE goals even to the extent of increasing the size of the weapon arsenal. This subroutine also forces the user to select an allocation that minimizes one of the following:

- Warheads
- Megatonnage
- CMP
- EMP

PAGE -- Pages the display to the screen

PARCH1 -- Called when the user wants to change one of the following target parameters:
- Name
- Vulnerability
- Category
- Number of targets in the class
- Diameter
- Priority
- Value

PARCH2 -- Called when the user wants to change one of the following force target parameters:
- Reliability
- Cep
- Yield
- Warheads/weapon system

PERM -- Performs the pivot operation on the pivot element

PHSE1 -- Reads in any real constraints and performs a simplex procedure to find an initial basic feasible solution

PK -- (Function) Calculates the single shot probability of survival for each weapon/target pairing if PSI expresses the target vulnerability

PLACE -- Puts the objective function weights for the deviation variables at the current priority in the correct position in the tableau

POUT -- Prepares and prints the solution information

READ1 -- Reads in the goal constraints and objective function terms assigned to priority one

READ2 -- Reads in the goal constraints and objective function terms assigned to the current priority

TEST -- Determines the next entering variable's column and row.

TGTEDT -- Allows the user to edit his target parameters and to add or delete a target class

TGTEXC -- Allows the user to replace a target class with another class

TGTFIL -- Writes all target class data to an external formated file

TGTINF -- Reads t rget data from an external user-defined formated file

TGTINS -- Allows the user to interactively enter target class data

TGTP1 -- Displays the data that PARCH1 manipulates on the screen

TGTP2 -- Displays the data that PARCH2 manipulates on the screen

VTK -- (Function) Computes single shot probability of survival if a VNTK number expresses target vulnerability

WEIGHT -- Allows the user to correct or change the value of any target class

WPNCH1 -- Allows the user to change the following weapon class data:
- Number of weapons
- Number of warheads/weapon
- Reliability
- CEP
- Yield

WPNCH2 -- Allows the user to change the following weapon class data:
- Daily alert rate
- Generated alert rate

WPNEDT -- Allows the user to edit his weapon parameters and to either add or delete a weapon class

WPNEXC -- Allows the user to replace a target class with another class

WPNFIL -- Writes all weapon class data to an external file

WPNINF -- Reads weapon data from an external file

WPNINS -- Allows the user to interactively enter weapon class data

WPNP1 -- Displays to the monitor the data that WPNCH1 manipulates

WPNP2 -- Displays to the monitor the data that WPNCH2 manipulates

WTINTR -- Allows the user to specify which weapon/target assignments are inappropriate

ZEROIZ -- Initializes all variables and matrices

APPENDIX C

BRIK Variable Listing

VARIABLE	DEFINITION
achiev(j)	Damage Expectancy achievement of the target class
aij(j,i)	Coefficients of the aij matrix for row j and column i, RHS is located in column 1
AIJ2	File for aij when problem is rerun
alloc	File containing allocation output
cr	Crater radius
dl	Lethal diameter
emt	Equivalent megatonnage
hedglt	Number of available hedging constraints
iconnp(j,n)	Subscript of the jth constraint at priority n
icount	The ith run of the problem
IND(I)	Indicator row that marks the eligibility of a variable to enter the basis
ipcnt	Counter used to page the screen
iprin(i)	Priority of the ith constraint
isub(i,n)	Subscript of the nth constraint at priority i
itype(i,n)	The type of the nth deviational variable at priority i, 3, Positive deviational variable 4, Negative deviational variable
JCOL(i,1)	Type of variable in column i

JCOL(i,2)	Subscript of the variable in column i
JROW(i,1)	The type of basic variable in row i, where type is: X = 2; p = 3; n = 4
JROW(i,2)	Subscript of the basic variable in row i
mtgts	The amount of additional target classes that can be added
mwpns	The amount of additional weapon classes that can be added
nc(n)	Number of constraints assigned to priority n
NCOLI	Number of columns in the current tableau
ncon(j,n)	Subscript of the jth constraints at priority n
nhedg	Number of hedging constraints used in the current problem
node	Indicates a change has been made to the target class data
NPRIC	Priority currently being optimized
nprit	Total number of priorities in problem
nrcon	Number of real constraints
NROWI	Number of rows in the current working tableau
ntgtpr(i)	Priority of target class i (1 - 7)
ntgts	Number of target classes
ntgtwh(i)	Number of warheads for a force type target i

ntof(n)	Number of terms in the objective function for priority n
numcon	Number of constraints
numtgt(i)	Number of targets in target class i
numwpn(j)	Number of weapon systems in weapon class j
numwrh(i)	Number of warheads/weapon system in weapon class i
nvar	Number of decision variables
nwpns	Number of weapon classes
pk1	Correction factor applied to pk for area targets (Diameter>0)
rhs(i)	Right hand side of constraint i
rl	Lethal radius
sparse(i,j)	Single shot probability of survival for target class i against weapon class j
TA(N)	Total deviation from the goals at priority n
TB(i)	Right hand side constant of the constraint in row i
tcmp	Countermilitary potential
TE(i,j)	Coefficient of the variable in column j of the constraint in row i
tgtcat(i)	Category of target class i v=value target m=military target f=force target
tgtcep(i)	CEP of weapon in force target class i
tgtde(i)	Desired level of destruction to target class i (%)

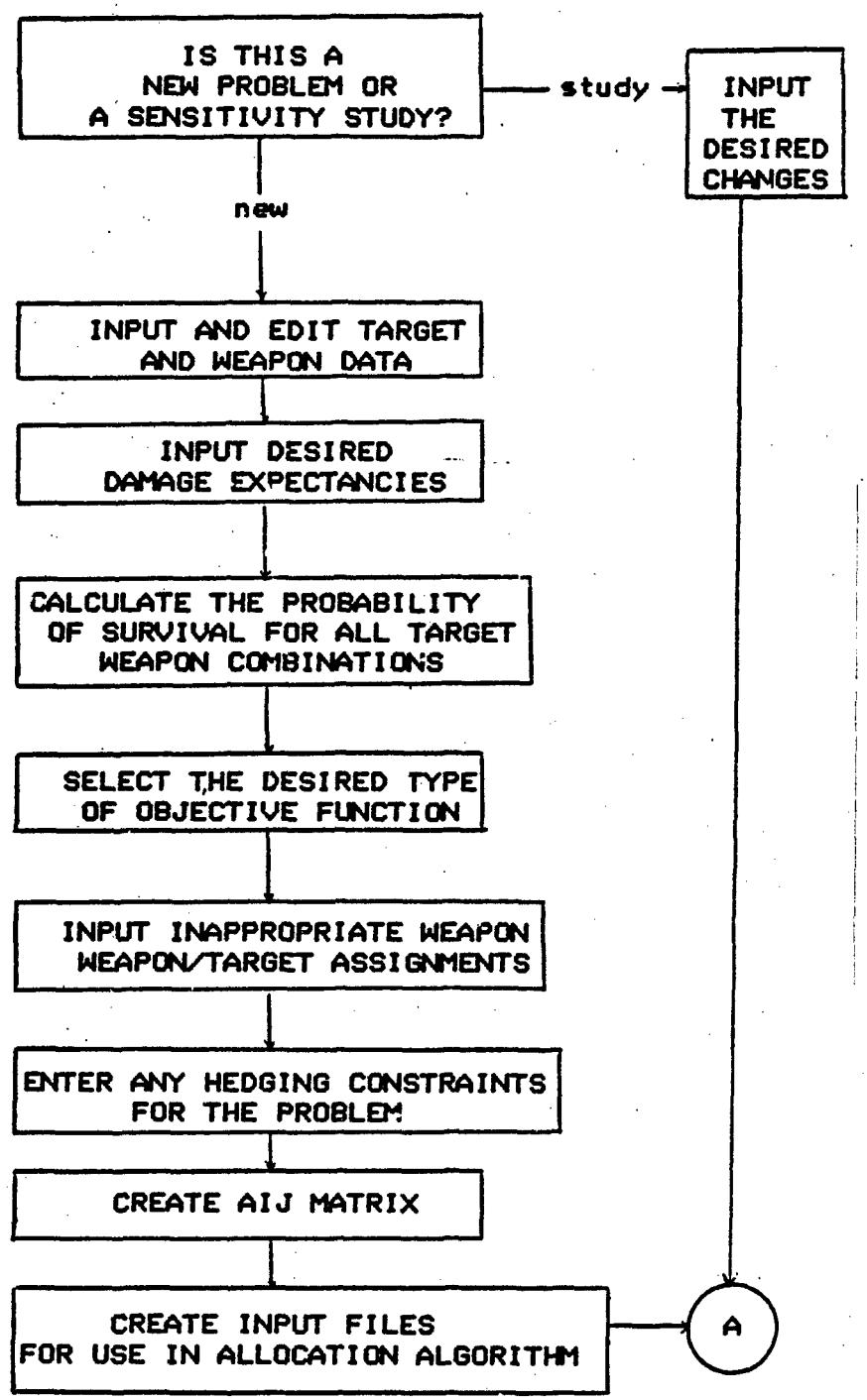
tgtdia(i)	Diameter (NM) of a target in class i
tgtnam(i)	Name of target class i
tgtpsi(i)	Vulnerability of target class i designated by a VNTK number or PSI hardness value
tgtrel(i)	Reliability of force target class i
tgtrmn(i)	Amount of targets remaining in class i after the allocation
tgtval(i)	Value of one target in target class i
tgtyld(i)	Yield of a weapon in force target class i
TL(I,N)	Weight assigned to the basic variable in row i at priority n
tpk	Probability of kill
TT(N,J)	Weight of the variable in column j at priority n
valdes	Total value destroyed by the allocation
wght(i,n)	Weight (Value) of the nth deviational variable at priority i
wleft(i)	Amount of unused weapons in class i after the allocation
wpname(j)	Name of weapon class j
wpncep(j)	CEP of a weapon warhead in weapon class j
wpndda(j)	Daily alert rate for weapon class j
wpnga(j)	Generated alert rate for weapon class j
wpnrel(j)	Reliability (Probability of

wpnrel(j)	Reliability (Probability of penetration) for a warhead in weapon class j
wpnyld(j)	Warhead yield for a weapon in class j
wused(i)	Amount of weapons in class i used in the allocation
xxx	Dummy variable which reads the target and weapon files header
yeswgt	Indicates if the program will compute values for force targets (0 = no and 1 = yes)

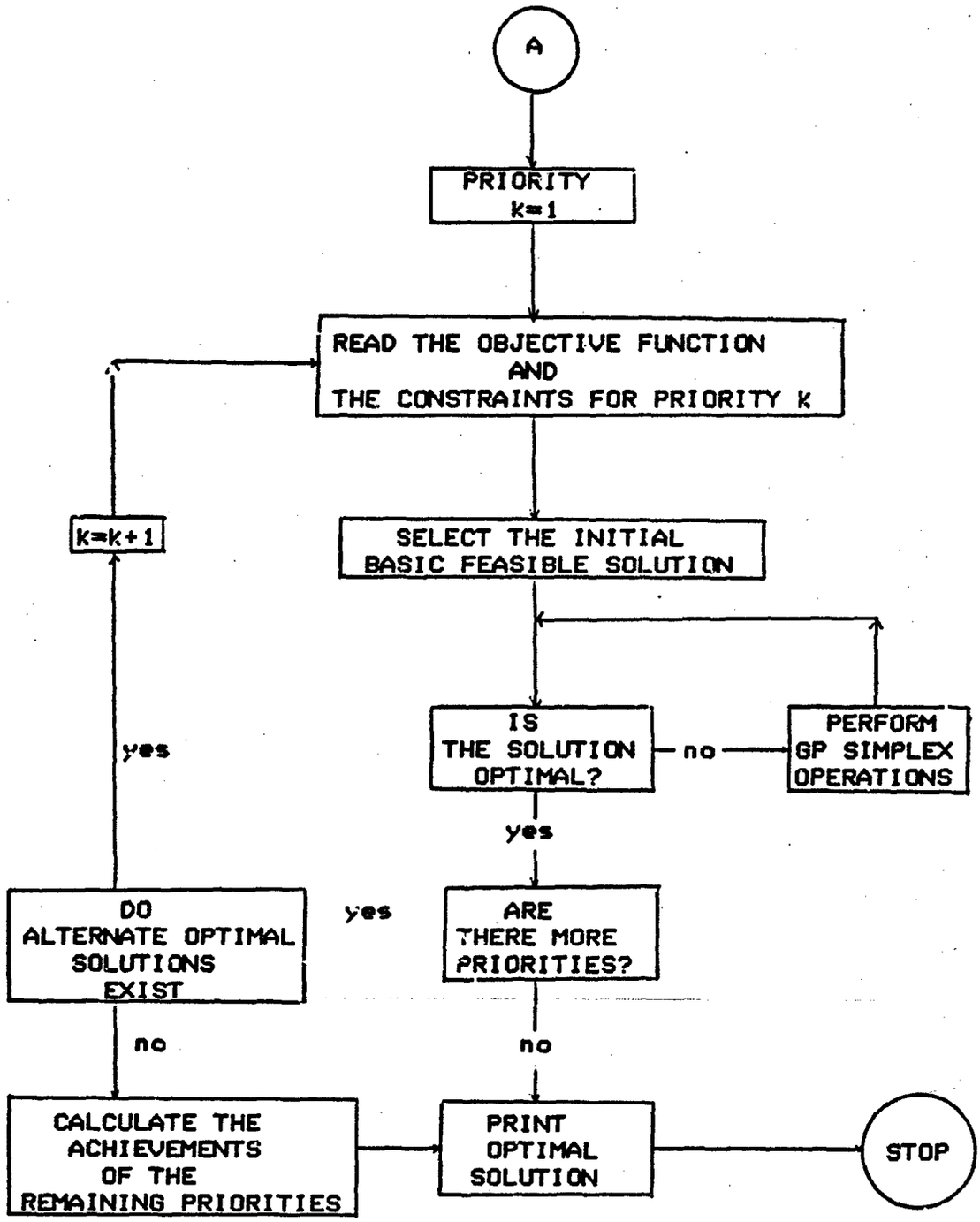
APPENDIX D

BRIK FLOWCHARTS

Flowchart for Input Section of BRIK



Flowchart for BRIK Allocation Algorithm



```
      program brik
      common/main/ntof(10),iprin(61),isub(10,20),itype(10,20),
     c             wght(10,20),wpnyld(20),tgtval(20),ntgtpr(20),
     c             rhs(61),nprit,nhedg,numtgt(20),wpncep(20)
      common/aijin1/nwpns,ntgts,sparse(20,20),aij(61,401),icount
      common/wtint/wpname(20),tgtnam(20)
      common/cat/tgtcat(20)
      dimension tgtdia(20),wpndda(20),wpnga(20),
     c          tgtrel(20),tgtcep(20),ntgtwh(20),numwrh(20),
     c          tgtyld(20),numwpn(20),wpnrel(20),tgtde(20)
      character wpname(20)*6,tgtnam(20)*6,tgtcat(20),yeswgt,
     c           tgtpsi(20)*4,t,vntK*4,a,b,c,d
      call header
      write(6,190)
400   write(6,*)'What type of analysis is this?  Please select one.'
      write(6,*)'  1)   New Analysis.'
      write(6,*)'  2)   Continuation of an existing analysis.'
      write(6,*)'  3)   Quit.'
2     read(5,*)ncont
      if (ncont.lt.1 .or. ncont.gt.3) go to 2
      if (ncont.eq.3) go to 500
      if (ncont.eq.2) then
            write(6,190)
            write(6,*)'                                        PLEASE STANDBY'
            write(6,195)
            call change
            go to 200
      endif
      yeswgt='n'
      node=0
      call zeroiz(numtgt,tgtpsi,tgtdia,ntgtpr,tgtrel,tgtcep,tgtyld,
     c            ntgtwh,tgtval,tgtde,numwpn,numwrh,wpncep,wpnrel,
     c            wpnyld,wpndda,wpnga,sparse,rhs,aij,isub,itype,wght,
     c            ntof,iprin,wpname,tgtnam)
      icount=1
      open(11,file='PAGP2')
      rewind 11
```

```
      write(11,1) icount,icount,icount
    1 format(3i5)
      close (11)
      write(6,*)
      write(6,*)'How will you enter your target data?  Please select o
     1ne.'
      write(6,*)'  1)   Interactively.'
      write(6,*)'  2)   From a user designated file.'
    3 read(5,*)nfile
      if (nfile.ne.1 .and. nfile.ne.2) go to 3
      if (nfile.eq.1) then
            call tgtins(ntgts,numtgt,tgtpsi,tgtdia,ntgtpr,
     c                  tgtrel,tgtcep,tgtyld,ntgtwh,tgtval,tgtde,
     c                  tgtnam,tgtcat,yeswgt,node)
      else
            call tgtinf(ntgts,numtgt,tgtpsi,tgtdia,ntgtpr,
     c                  tgtrel,tgtcep,tgtyld,ntgtwh,tgtval,tgtde,
     c                  tgtnam,tgtcat,yeswgt,node)
      endif
      call tgtfil(ntgts,numtgt,tgtpsi,tgtdia,ntgtpr,tgtrel,
     c            tgtcep,tgtyld,ntgtwh,tgtval,tgtnam,tgtcat,node)
      write(6,190)
      write(6,*)
      write(6,*)'How will you enter your weapon data?  Please select o
     1ne.'
      write(6,*)'  1)   Interactively.'
      write(6,*)'  2)   From a user designated file.'
    4 read(5,*)nfile
      if (nfile.ne.1 .and. nfile.ne.2) go to 4
      if (nfile.eq.1) then
            call wpnins(nwpns,numwpn,numwrh,wpncep,wpnrel,wpnyld,
     c                  wpndda,wpnga,wpname)
      else
            call wpninf(nwpns,wpname,numwpn,
     c                  numwrh,wpncep,wpnrel,wpnyld,wpndda,wpnga)
      endif
      call wpnfil(nwpns,numwpn,numwrh,wpncep,
     c            wpnrel,wpnyld,wpndda,wpnga,wpname)
```

```
            write(6,190)
      c
      c     DE data must be entered manually if tgt files were changed.
      c
            if (node.eq.0) then
            write(6,*)'How will you enter your DE goals?  Please select one.'
            write(6,*)'     1)  Interactively.'
            write(6,*)'     2)  A user defined file.'
      5     read(5,*)nfile1
            if (nfile1.ne.1 .and. nfile1.ne.2) go to 5
            if (nfile1.eq.2) then
                    call defile(tgtde,ntgts,tgtnam,tgtcat)
            else
                    call deinac(tgtde,ntgts,tgtcat,tgtnam)
            endif
            else
            call deinac(tgtde,ntgts,tgtcat,tgtnam)
            endif
            write(6,190)
            write(6,*)'                                          PLEASE STANDBY'
            write(6,195)
      c
      c
      c
      c
      c*******Computation of SSK for Sparse matrix
      c
      c
            open (15,file='vntkch')
      c
      c
      c
            do 50 i=1,ntgts
                            rewind 15
                            vntK=tgtpsi(i)
                            write(15,196)vntK
                            rewind 15
                            read(15,197)a,b,c,d
```

```
                        rewind 15
                        if (c.eq.'q' .or. c.eq.'Q' .or. c.eq.'p' .or.
     c                  c.eq.'P') then
                         read(15,198)in,t,ik
                         vn=real(in)
                         r95=tgtdia(i)/2
                         do 60 j=1,nwpns
                         tpk=vtk(vn,t,ik,r95,wpnyld(j),wpncep(j))
                         sparse(i,j)=1-wpnrel(j)*tpk
60                       if (sparse(i,j).lt..001) sparse(i,j)=.001
                        else
                         read(15,199)npsi
                         psi=real(npsi)
                         do 61 j=1,nwpns
                         tpk=pk(wpnyld(j),psi,wpncep(j),tgtdia(i))
                         sparse(i,j)=1-wpnrel(j)*tpk
61                       if (sparse(i,j).lt..001) sparse(i,j)=.001
                        endif
50      continue
        close (15)
        write(6,190)
        write(6,*)'What type of alert status is the weapon base on?'
        write(6,*)'1)  Day-to-Day'
        write(6,*)'2)  Generated '
        write(6,*)'Select one'
65      read(5,*)nalert
        if (nalert.ne.1 .and. nalert.ne.2) go to 65
        do 70 i=1,ntgts
70              rhs(i)= (-1)*numtgt(i)*log(1-tgtde(i))
        do 80 i=1,nwpns
                if (nalert.eq.1) then
                        rhs(ntgts+i)=numwpn(i)*wpndda(i)*numwrh(i)
                else
                        rhs(ntgts+i)=numwpn(i)*wpnga(i)*numwrh(i)
                endif
80      continue
        open(11,file='cdat')
        rewind 11
```

```
      do 90 j=1,ntgts
         write(11,480) tgtpsi(j),tgtdia(j),tgtde(j),tgtcat(j)
90    continue
      do 95 i=1,nwpns
         write(11,*) wpnrel(i),wpncep(i),wpnyld(i)
95    continue
      close (11)
      call object(tgtde)
      call wtintr
      call hedge
      call aijin
      call filein
200   write(6,190)
      call bout
      write(6,190)
      go to 400
190   format(////////////////////////)
195   format(//////////)
196   format(a4)
197   format(4a1)
198   format(i2,a1,i1)
199   format(i4)
480   format(a4,f7.3,f4.2,a1)
500   stop
      end
c
c
c
      subroutine zeroiz(numtgt,tgtpsi,tgtdia,ntgtpr,tgtrel,tgtcep,
     c                  tgtyld,ntgtwh,tgtval,tgtde,numwpn,numwrh,
     c                  wpncep,wpnrel,wpnyld,wpndda,wpnga,sparse,
     c            rhs,aij,isub,itype,wght,ntof,iprin,wpname,tgtnam)
      dimension numtgt(20),tgtdia(20),ntgtpr(20),tgtrel(20),
     c          tgtcep(20),tgtyld(20),ntgtwh(20),tgtval(20),tgtde(20),
     c        numwpn(20),numwrh(20),wpncep(20),wpnrel(20),wpnyld(20),
     c        wpndda(20),wpnga(20),sparse(20,20),rhs(61),aij(61,401),
     c        isub(10,20),itype(10,20),wght(10,20),ntof(10),iprin(61)
      character wpname(20)*6,tgtnam(20)*6,tgtpsi(20)*4
```

```
      do 10 i=1,20
              numtgt(i)=0
              tgtpsi(i)='     '
              tgtdia(i)=0.
              ntgtpr(i)=0
              tgtrel(i)=0.
              tgtcep(i)=0.
              tgtyld(i)=0.001
              ntgtwh(i)=0
              tgtval(i)=1.
              tgtde(i)=0.
              numwpn(i)=0
              numwrh(i)=0
              wpncep(i)=0.
              wpnrel(i)=0.
              wpnyld(i)=0.
              wpndda(i)=0.
              wpnga(i)=0.
              wpname(i)='     '
              tgtnam(i)='     '
10    continue
      do 20 i=1,10
              ntof(i)=0
              do 30 j=1,20
                      isub(i,j)=0
                      itype(i,j)=0
                      wght(i,j)=0.
30            continue
20    continue
      do 35 i=1,20
        do 37 j=1,20
              sparse(i,j)=0.0
37      continue
35    continue
      do 40 i=1,61
              iprin(i)=0
              rhs(i)=0
              do 50 j=1,401
```

```
                        aij(i,j)=0.
50                  continue
40          continue
            return
            end


            subroutine tgtins(ntgts,numtgt,tgtpsi,tgtdia,ntgtpr,
           c                  tgtrel,tgtcep,tgtyld,ntgtwh,tgtval,tgtde,
           c                  tgtnam,tgtcat,yeswgt,node)
            dimension  numtgt(20),tgtdia(20),ntgtpr(20),
           c           tgtrel(20),tgtcep(20),tgtyld(20),ntgtwh(20),
           c           tgtval(20),tgtde(20)
            character yeswgt,val,tgtnam(20)*6,tgtcat(20),tgtpsi(20)*4
            character nam*6,ab
            write(6,180)
           write(6,*)'The following data is needed for each target class.'
4           format (a1)
            write(6,*)
            write(6,*)'          TARGET CLASS'
            write(6,*)
            write(6,*)'Class name                -          - max 6 characters'
            write(6,*)'Number in the class       -          - integer'
            write(6,*)'Target vulnerability      -    VNTK - 4 characters'
            write(6,*)'                          -     PSI - integer'
            write(6,*)'Target diameter           -      NM - real'
            write(6,*)'Target category           -   FORCE - f '
            write(6,*)'                          -   VALUE - v '
            write(6,*)'                          -   OTHER - m '
            write(6,*)'Target class priority -- 1 to 7 -- integer'
            write(6,*)
            write(6,*)'If you do not have the above data, type 1 and the '
            write(6,*)'program will terminate, otherwise hit return.'
            write(6,180)
            read(5,4)ab
            if (ab.eq.'1')stop
            write(6,190)
            write(6,*)'This program can also calculate force target values.'
```

```
      write(6,*)'The following data is needed:'
      write(6,*)
      write(6,*)'Target reliability--  % -- real'
      write(6,*)'CEP                -- NM -- real'
      write(6,*)'Yield per warhead  -- MT -- real'
      write(6,*)'Warheads per tgt   --    -- integer'
      write(6,*)
      write(6,*)
     write(6,*)'Do you want the program to calculate the force target'
      write(6,*)'values? (y/n)'
      write(6,180)
5     format(a1)
6     format(a4)
7     format(a6)
10    read(5,5)yeswgt
      if (yeswgt.ne.'y' .and. yeswgt.ne.'n') go to 10
      write(6,190)
      write(6,*)'I am ready for you to input the target class data.'
      ntgts=0
      i=0
100   write(6,*)
      write(6,*)'Input a target name.  Input 0 if finished.'
      read(5,7)name
      if (name.eq.'0') go to 110
      i=i+1
      tgtnam(i)=name
      write(6,*)'Input the target category (m,v,f).'
101   read(5,5)tgtcat(i)
      if(tgtcat(i).ne.'v'.and.tgtcat(i).ne.'f'.and.tgtcat(i).ne.'m')
     1  go to 101
     write(6,*)'Input the target NTK (4 characters) or PSI (Integer)'
      write(6,*)'number.'
      read(5,6)tgtpsi(i)
      if (yeswgt.eq.'y' .and. tgtcat(i).eq.'f') then
              write(6,*)'Input the following data in order:'
              write(6,*)'Number of targets, target diameter (NM),
     1 priority (1-7), reliability, cep (NM),'
              write(6,*)'yield (MT), warheads per weapon.'
```

```
                read(5,*)numtgt(i),tgtdia(i),ntgtpr(i),
     c                     tgtrel(i),tgtcep(i),tgtyld(i),ntgtwh(i)
        else
                write(6,*)'Input the following:'
                write(6,*)'Number of targets, target diameter(NM), prio
     crity (1-7).'
                read(5,*)numtgt(i),tgtdia(i),ntgtpr(i)
        endif
        ntgts=ntgts+1
        if (ntgts.eq.20) then
             write(6,*)'You have used up your tgt class memory space.'
             go to 110
        endif
        write(6,190)
        go to 100
110     write(6,190)
        if (yeswgt.eq.'n') then
                write(6,*)'Do you want to assign values to each target
     c in a class? (y/n)'
                write(6,*)'This program defaults to 1 unit of value for
     c each target.'
111             read(5,5)val
                if (val.ne.'y' .and. val.ne.'n') go to 111
                if(val.eq.'y') then
                  write(6,190)
                  write(6,*)'After I display the target class name, inp
     lut your target class value.'
                do 112 i=1,ntgts
                    write(6,*)
                    write(6,7)tgtnam(i)
112                 read(5,*)tgtval(i)
                endif
        else
                write(6,*)'Do you want to assign values to your value or
     c military targets? (y/n)'
                write(6,*)'This program defaults to one unit of value fo
     1r each target.'
113             read(5,5)val
```

```
                if (val.ne.'y' .and. val.ne.'n') go to 113
                if (val.eq.'y') then
                    write(6,*)'After I display the target class name, in
     cput your target class value.'
                    do 114 i=1,ntgts
                        write(6,*)
                        if (tgtcat(i).ne.'f') then
                           write(6,*)tgtnam(i)
                           read(5,*)tgtval(i)
                        endif
114                 continue
                endif
            endif
            write(6,190)
            call tgtedt(yeswgt,ntgts,numtgt,tgtpsi,tgtdia,ntgtpr,
     c                  tgtrel,tgtcep,tgtyld,ntgtwh,tgtval,tgtde,
     c                  tgtnam,tgtcat,node)
            do 160 i=1,ntgts
                if (ntgtpr(i).lt.1) ntgtpr(i)=1
                if (ntgtpr(i).gt.7) ntgtpr(i)=7
                if (numtgt(i).lt.0) numtgt(i)=0
                if (tgtcat(i).ne.'v' .and. tgtcat(i).ne.'f' .and.
     c                  tgtcat(i).ne.'m') tgtcat(i)='v'
                if (tgtrel(i).lt.0) tgtrel(i)=.000001
                if (tgtrel(i).gt.1.) tgtrel(i)=1.
                if (ntgtwh(i).lt.0) ntgtwh(i)=0
                if (tgtcep(i).le.0.) tgtcep(i)=.000001
                if (tgtyld(i).lt.0.) tgtyld(i)=.000001
160             if (tgtval(i).lt.0.) tgtval(i)=.000001
            return
180         format(///////)
190         format(////////////////////////)
            end
c
c
            subroutine wpnins(nwpns,numwpn,
     c                  numwrh,wpncep,wpnrel,wpnyld,wpndda,wpnga,wpname)
            dimension  numwpn(20),wpnrel(20),wpncep(20),
```

```
c              wpnyld(20),wpndda(20),wpnga(20),numwrh(20)
      character wpname(20)*6
      character name*6,ab
      write(6,180)
     write(6,*)'The following data is needed for each weapon class.'
4     format (a1)
7     format (a6)
      write(6,*)
      write(6,*)'      WEAPON CLASS DATA'
      write(6,*)
      write(6,*)'Class name              --    -- max 6 characters'
      write(6,*)'Number in class         --    -- integer'
      write(6,*)'Warheads per weapon     --    -- integer'
      write(6,*)'Reliability             --  % -- real'
      write(6,*)'Weapon CEP              -- NM -- real'
      write(6,*)'Weapon yield            -- MT -- real'
      write(6,*)'Daily  alert rate       --  % -- real'
      write(6,*)'Generated Alert rate    --  % -- real'
      write(6,*)
      write(6,*)'If you do not have the data, type 1 and the program'
      write(6,*)'will terminate, otherwise hit return.'
      write(6,180)
      read(5,4)ab
      if (ab.eq.'1')stop
      write(6,*)
      write(6,190)
      write(6,*)'I am ready for you to input the weapon class data.'
      nwpns=0
      i=0
120   write(6,*)'Enter the weapon class name.  Enter 0 if finished.'
      read(5,7)name
      i=i+1
      if (name.eq.'0') go to 150
      wpname(i)=name
      write(6,*)'Input the following data:'
      write(6,*)'number of weapons, warheads per weapon, reliability,'
      write(6,*)'CEP (NM), yield (MT), daily alert rate, generated'
      write(6,*)'alert rate.'
```

```
      read(5,*)numwpn(i),numwrh(i),wpnrel(i),wpncep(i),wpnyld(i),
     c   wpndda(i),wpnga(i)
      nwpns=i
      if (nwpns.eq.20) then
      write(6,*)'You have filled your weapon class storage space.'
            go to 150
      endif
      write(6,190)
      go to 120
150   call wpnedt(nwpns,wpnga,wpname,numwpn,
     c            numwrh,wpncep,wpnrel,wpnyld,wpndda)
      do 170 i=1,nwpns
            if (numwpn(i).lt.0) numwpn(i)=0
            if (numwrh(i).lt.0) numwrh(i)=0
            if (wpnrel(i).lt.0) wpnrel(i)=.000001
            if (wpnrel(i).gt.1.) wpnrel(i)=1.
            if (wpncep(i).le.0.) wpncep(i)=.00001
            if (wpnyld(i).lt.0.) wpnyld(i)=.00001
            if (wpndda(i).lt.0.) wpndda(i)=.00001
            if (wpndda(i).ge.1.) wpndda(i)=1.
            if (wpnga(i).lt.0.) wpnga(i)=.00001
170         if (wpnga(i).ge.1.) wpnga(i)=1.
      return
180   format(//////)
190   format(////////////////////////)
      end
c
c
      subroutine tgtinf(ntgts,numtgt,tgtpsi,tgtdia,
     c      ntgtpr,tgtrel,tgtcep,tgtyld,ntgtwh,tgtval,tgtde,
     c      tgtnam,tgtcat,yeswgt,node)
      character tgtcat(20),tgtnam(20)*6,name*6,yeswgt,xxx*59
      character tgtpsi(20)*4
      dimension numtgt(20),tgtdia(20),ntgtpr(20),
     c          tgtrel(20),tgtcep(20),tgtyld(20),tgtval(20),
     c          tgtde(20),ntgtwh(20)
      write(6,*)'Enter your file that has your target data.'
2     format(a6)
```

```
      read(5,2)name
      open(10,file=name)
      rewind 10
      read(10,*)ntgts
5     format(a59)
7     format(a6,x,i4,x,a4,x,f5.2,3x,a1,4x,i1,2x,f5.2,x,f4.2,x,f4.2,x,
     1       f5.2,2x,i2)
      read(10,5)xxx
      do 10 i=1,ntgts
          read(10,7)tgtnam(i),numtgt(i),tgtpsi(i),tgtdia(i),tgtcat(i),
     c    ntgtpr(i),tgtval(i),tgtrel(i),tgtcep(i),tgtyld(i),ntgtwh(i)
10    continue
      close (10)
      write(6,*)
      call tgtedt(yeswgt,ntgts,numtgt,tgtpsi,tgtdia,ntgtpr,
     c            tgtrel,tgtcep,tgtyld,ntgtwh,tgtval,tgtde,
     c            tgtnam,tgtcat,node)
      do 30 i=1,ntgts
            if (ntgtpr(i).lt.1) ntgtpr(i)=7
            if (ntgtpr(i).gt.7) ntgtpr(i)=7
            if (numtgt(i).lt.0) numtgt(i)=0
            if (tgtcat(i).ne.'f' .and. tgtcat(i).ne.'v' .and.
     c         tgtcat(i).ne.'m') tgtcat(i)='v'
            if (tgtdia(i).lt.0.) tgtdia(i)=.00001
            if (tgtrel(i).lt.0.) tgtrel(i)=.00001
            if (tgtrel(i).gt.1.) tgtrel(i)=1.
            if (tgtcep(i).le.0.) tgtcep(i)=.00001
            if (tgtyld(i).lt.0.) tgtyld(i)=.00001
            if (ntgtwh(i).lt.0) ntgtwh(i)=0
30          if (tgtval(i).lt.0.) tgtval(i)=.00001
45    format(a1)
      return
      end
c
c
      subroutine wpninf(nwpns,wpname,
     c      numwpn,numwrh,wpncep,wpnrel,wpnyld,wpndia,wpnga)
      character wpname(20)*6,name*6,xxx*56
```

```
      dimension wpnga(20),numwpn(20),wpnrel(20),
     c          numwrh(20),wpncep(20),wpnyld(20),wpndda(20)
5     format (a6,2x,i5,4x,i3,2x,f5.2,x,f5.3,x,f5.2,3x,f4.2,5x,f4.2)
6     format(a56)
      write(6,*)'Enter your file that has your weapon data.'
      read(5,5)name
      open(10,file=name)
      rewind 10
      read(10,*)nwpns
      read(10,6)xxx
      do 20 i=1,nwpns
          read(10,5)wpname(i),numwpn(i),numwrh(i),wpnrel(i),wpncep(i),
     c              wpnyld(i),wpndda(i),wpnga(i)
20    continue
      close (10)
      write(6,*)
      call wpnedt(nwpns,wpnga,wpname,numwpn,
     c            numwrh,wpncep,wpnrel,wpnyld,wpndda)
      do 40 i=1,nwpns
            if (numwpn(i).lt.0) numwpn(i)=0
            if (numwrh(i).lt.0) numwrh(i)=0
            if (wpnrel(i).lt.0.) wpnrel(i)=.00001
            if (wpnrel(i).gt.1.) wpnrel(i)=1.
            if (wpncep(i).le.0.0) wpncep(i)=.00001
            if (wpnyld(i).lt.0.) wpnyld(i)=.00001
            if (wpndda(i).lt.0.) wpndda(i)=.00001
            if (wpndda(i).ge.1.) wpndda(i)=1.
            if (wpnga(i).lt.0.) wpnga(i)=.00001
40          if (wpnga(i).ge.1.) wpnga(i)=1.
45    format(a1)
      return
      end
c
c
      subroutine tgtfil(ntgts,numtgt,tgtpsi,tgtdia,ntgtpr,
     c    tgtrel,tgtcep,tgtyld,ntgtwh,tgtval,tgtnam,tgtcat)
      character tgtnam(20)*6,tgtcat(20),name*6,tgtpsi(20)*4
      dimension numtgt(20),tgtdia(20),ntgtpr(20),
```

```
      c              tgtrel(20),tgtcep(20),tgtyld(20),ntgtwh(20),
      c              tgtval(20)
            write(6,310)
5           format(a6)
7           format(a6,x,i4,x,a4,x,f5.2,3x,a1,4x,i1,2x,f5.2,x,f4.2,x,f4.2,x,
     1           f5.2,2x,i2)
8           format(' NAME  NUMB   VUL   DIA TYPE PRIO VAL   REL  CEP  YLD  W
     1HDS')
            write(6,*)'You can save your target  data'
            write(6,*)' 1)  on a new file,'
            write(6,*)' 2)  over an old file,'
            write(6,*)' 3)  not save the data.'
            write(6,*)'Select one.'
20          read(5,*)nsave1
            if (nsave1.lt.1 .or. nsave1.gt.3) go to 20
            go to (100,100,300)nsave1
100         write(6,*)
            write(6,*)' What is your file name? (Max 6 characters)'
            read(5,5)name
            open (10,file=name)
            rewind 10
            write(10,*)ntgts
            write(10,8)
            do 105 i=1,ntgts
               write(10,7)tgtnam(i),numtgt(i),tgtpsi(i),tgtdia(i),tgtcat(i),
      c         ntgtpr(i),tgtval(i),tgtrel(i),tgtcep(i),tgtyld(i),ntgtwh(i)
105         continue
            close (10)
300         return
310         format(//////////////////////)
            end
      c
      c


            subroutine wpnfil(nwpns,numwpn,numwrh,wpncep,
      c        wpnrel,wpnyld,wpndda,wpnga,wpname)
            character wpname(20)*6,name*6
            dimension numwpn(20),wpnrel(20),wpncep(20),
```

```
c             wpnyld(20),wpndda(20),wpnga(20),numwrh(20)
      write(6,310)
5     format (a6,2x,i5,4x,i3,2x,f5.2,x,f5.3,x,f5.2,3x,f4.2,5x,f4.2)
6     format('NAME  NUMWPNS  WHD/WPN  REL  CEP   YLD  DAYALRT  GENALRT
     1')
      write(6,*)'You can save your weapon data'
      write(6,*)' 1)  on a new file,'
      write(6,*)' 2)  over an old file,'
      write(6,*)' 3)  not save the data.'
      write(6,*)'Select one.'
20    read(5,*)nsave1
      if (nsave1.lt.1 .or. nsave1.gt.3) go to 20
      go to (100,100,300)nsave1
100   write(6,*)
      write(6,*)' What is your file name? (Max 6 characters)'
      read(5,5)name
      open (10,file=name)
      rewind 10
      write(10,*)nwpns
      write(10,6)
      do 200 i=1,nwpns
         write(10,5)wpname(i),numwpn(i),numwrh(i),wpnrel(i),wpncep(i),
c                wpnyld(i),wpndda(i),wpnga(i)
200   continue
      close (10)
300   return
310   format(//////////////////////)
      end
c
c
c
      subroutine calwgt(ntgts,ntgtwh,tgtval,tgtrel,
c        tgtyld,tgtcep,tgtcat)
      dimension ntgtwh(20),tgtval(20),tgtrel(20),
c        tgtyld(20),tgtcep(20)
      character tgtcat(20)
      do 20 i=1,ntgts
            if (tgtcat(i).eq.'f') then
```

```
            if(tgtcep(i).lt.,001) tgtcep(i)=,001
     tgtval(i)=tgtrel(i)*ntgtwh(i)*(tgtyld(i)**(2/3))/tgtcep(i)**2
            endif
20      continue
        close (18)
        return
        end
c
c
c
        subroutine weight(tgtval,tgtnam,ntgts)
        dimension tgtval(20)
        character tgtnam(20)*6,yes
1       format(a1)
5       write(6,35)
        write(6,*)'The following target values have been assigned.'
        write(6,*)
        write(6,*)' NO  TARGET CLASS NAME        TARGET VALUE'
        write(6,*)
10      format(x,i2,8x,a6,14x,f8.2)
        do 20 i=1,ntgts
20              write(6,10)i,tgtnam(i),tgtval(i)
        write(6,*)'Is this data O.K.? (y/n)'
25      read(5,1)yes
        if (yes.ne.'y' .and. yes.ne.'n') go to 25
        if (yes.eq.'y') return
        write(6,*)
        write(6,*)'Enter the target number and the new value.'
        write(6,*)'When finished input 0,0.'
30      read(5,*)i,wght
        if (i.lt.0 .or. i.gt.ntgts) go to 30
        if (i.eq.0) go to 5
        tgtval(i)=wght
        go to 30
35      format(////////////////////)
        end
c
c
```

```
c
      function pk(yield,psi,cep,diam,rel)
      c1=7.45995
      c2=.207535
      c3=2.42620
      c4=5.89243
      c5=.0930345
      c6=3.83323
      c7=1.68900
      c8=1.09218
      c9=-(.29698)
      c10=2.73109
      c11=1.12109
      c12=.790655
      c13=.638752
      if (psi.le.10.0) then
            rl=(6.81*yield**(2./3.))/psi**.62
      else
            rl=2.8*yield**(1./3.)*((psi-7.37)**(-.352))
      endif
      cr=.125*yield**(.3)
      if (cr.gt.rl) rl=cr
      pk=1 - .5**((rl/cep)**2)
      if (diam.lt..0001) go to 20
      dl=2*rl
      y=rl/cep
      if (y.le.1.0) then
            cc=c1*c2**y*y**c3
      else
            cc=1+c4*c5**y*y**c6
      endif
      if (y.le.2.0) then
            b=c7*c8**y*y**c9
      else
            b=c10-c11*c12**y*y**c13
      endif
      x=(dl/diam)**2
      if (x.ge.cc) go to 20
```

```
      y=x/cc
      pk1=(y*(1-(1-y)**b)+(1-y)*(y)**(1/b))
      pk=pk*pk1
 20   return
      end
c
c
      subroutine defile(tgtde,ntgts,tgtnam,tgtcat)
      dimension tgtde(20)
      character name*6,yes,tgtnam(20)*6,tgtcat(20)
 5    format(a6)
      write(6,*)
      write(6,*)'What file holds your DE data? (must be < 6 '
      write(6,*)'characters)'
      write(6,*)
      read(5,5)name
      open (12,file=name)
      rewind 12
      do 10 i=1,ntgts
 10   read (12,*)tgtde(i)
      write(6,*)
      write(6,*)'Do you want to check or change any DE data?'
      write(6,*)'(y/n)'
 15   format(a1)
 20   read(5,15)yes
      if (yes.ne.'y' .and. yes.ne.'n') go to 20
      if (yes.eq.'y') then
         call dechek(tgtde,tgtnam,ntgts,tgtcat)
         call desave(ntgts,tgtde)
      endif
      return
      end
c
c
      subroutine dechek(tgtde,tgtnam,ntgts,tgtcat)
      dimension tgtde(20)
      character yes,tgtnam(20)*6,tgtcat(20),ab
 2    write(6,40)
```

```
      write(6,*)' No      Tgt name   Tgt Cat      Tgt De'
      write(6,*)
5     format(x,i2,7x,a6,9x,a1,6x,f6.2)
      do 10 i=1,ntgts
      write(6,5)i,tgtnam(i),tgtcat(i),tgtde(i)
10    continue
      write(6,*)
      write(6,*)'Is the data O.K.? (y/n)'
12    read(5,15)yes
      if (yes.ne.'y' .and. yes.ne.'n') go to 12
      if (yes.eq.'y') return
15    format(a1)
      write(6,*)
      write(6,*)'Do you want to change DE by:'
      write(6,*)'    1)  Individual Target Class '
      write(6,*)'    2)  Target Category?'
      write(6,*)'Choose one.'
16    read(5,*)x
      if (x.ne.1 .and. x.ne.2) go to 16
      if (x.eq.1) then
      write(6,*)
      write(6,*)'Enter the number of the class that needs its DE '
      write(6,*)'changed.  If finished type 0.'
20    read(5,*)i
      if (i.lt.0 .or. i.gt.ntgts) go to 20
      if (i.eq.0) go to 2
      write(6,*)'Enter the DE.'
25    read(5,*)tgtde(i)
      if (tgtde(i).ge.1.0 .or. tgtde(i).lt.0.) then
            write(6,*)' Enter a DE between 0 and 1.'
            go to 25
      endif
      write(6,*)
      write(6,*)'Enter another class number.  Type 0 if finished.'
      go to 20
      else
      write(6,*)
      write(6,*)'Enter the category that needs changing. Enter 0 if
```

```
     1finished.'
30      read(5,15)ab
        if (ab.ne.'v' .and. ab.ne.'f' .and. ab.ne.'m' .and. ab.ne.'0')
     1              go to 30
        if (ab.eq.'0') go to 2
        write(6,*)
        write(6,*)'Enter the DE.'
35      read(5,*)de
        if (de.ge.1.0 .or. de.lt.0.) then
                write(6,*)' Enter a DE between 0 and 1.'
                go to 35
        endif
        do 36 i=1,ntgts
                if (tgtcat(i).eq.ab) tgtde(i)=de
36      continue
        write(6,*)
        write(6,*)'Enter another category.  Type 0 if finished.'
        go to 30
        endif
40      format(////////////////////)
        return
        end

        subroutine  deinac(tgtde,ntgts,tgtcat,tgtnam)
        dimension tgtde(20)
        character cata,yes,tgtcat(20),tgtnam(20)*6
        write(6,45)
        write(6,*)'Do you want to enter your DE data by:'
        write(6,*)'  1)  category (v=Value, m=Military, f=Force)'
        write(6,*)'  2)  individual class.'
        write(6,*)'Select one.'
5       read(5,*)n
        if (n.ne.1 .and. n.ne.2) go to 5
        if (n.eq.1) then
                do 10 i=1,3
                if (i.eq.1) cata='v'
                if (i.eq.2)cata='f'
                if (i.eq.3)cata='m'
```

```
6               write(6,*)'Enter your DE for your ',cata,' targets'
                read(5,*)de
                if (de.ge.1 .or. de.lt.0) then
                        write(6,*)'DE must be less than 1.'
                        go to 6
                endif
                write(6,*)
                do 20 j=1,ntgts
20                      if(cata.eq.tgtcat(j)) tgtde(j)=de
10              continue
        else
                do 25 i=1,ntgts
                write(6,40)tgtnam(i)
30              read(5,*)tgtde(i)
                if (tgtde(i).ge.1 .or. tgtde(i).lt.0) then
                        write(6,*)'DE must be less than 1.'
                        go to 30
                endif
25              continue
        endif
        write(6,*)
        write(6,*)'Do you want to check or change any DE data? (y/n)'
31      format(a1)
35      read(5,31)yes
        if (yes.ne.'y' .and. yes.ne.'n') go to 35
        if (yes.eq.'y') call dechek(tgtde,tgtnam,ntgts,tgtcat)
        call desave(ntgts,tgtde)
        return
40      format('Enter your DE for target class:  ',a6)
45      format(//////////////////////)
        end
c
c
c
        subroutine desave(ntgts,tgtde)
        dimension tgtde(20)
        character name*6
        write(6,400)
```

```
      write(6,*)'You can save your DE data'
      write(6,*)' 1)  on a new file,'
      write(6,*)' 2)  by replacing an old file,'
      write(6,*)' 3)  not save your new DE data.'
5     format(a6)
10    write(6,*)'Select one.'
      read(5,*)nsave
      if (nsave.lt.1 .or. nsave.gt.3) go to 10
      go to (100,100,300) nsave
100   write(6,*)
      write(6,*)' What is the file name? (Max 6 characters)'
      read(5,5)name
      open (12,file=name)
      rewind 12
      do 200 i=1,ntgts
      write(12,*)tgtde(i)
200   continue
      close (12)
300   return
400   format(//////////////////////)
      end

c
c
c
      subroutine tgtedt(yeswgt,ntgts,numtgt,tgtpsi,tgtdia,ntgtpr,
     c                  tgtrel,tgtcep,tgtyld,ntgtwh,tgtval,tgtde,
     c                  tgtnam,tgtcat,node)
      dimension numtgt(20),tgtdia(20),ntgtpr(20),tgtrel(20),
     c          tgtcep(20),tgtyld(20),ntgtwh(20),tgtval(20),tgtde(20)
      character yes,name*6,tgtnam(20)*6,tgtcat(20),yeswgt,tgtpsi(20)*4
      node=0
      write(6,*)'The following menu lists possible options to '
      write(6,*)'manipulate the data. (Hit return to continue)'
      read(5,4)yes
4     format(a1)
5     write(6,190)
      if (yeswgt.eq.'y') call calwgt(ntgts,ntgtwh,tgtval,tgtrel,
```

```
c                                       tgtyld,tgtcep,tgtcat)
      write(6,*)
      write(6,*)'                                MENU'
      write(6,*)
      write(6,*)'                          Target Class Category'
      write(6,*)
      write(6,*)'     1)      Check/View Data'
      write(6,*)'     2)      Change Parameters'
      write(6,*)'     3)      Add Class'
      write(6,*)'     4)      Replace Class'
      write(6,*)'     5)      Delete Class'
      write(6,*)
     write(6,*)'Enter your choice.  Type 0 if you do not want any of'
      write(6,*)'these options or are finished with your changes.'
6     format(a1)
7     format(a6)
10    read(5,*)nmenu
      if (nmenu.eq.0) return
      if (nmenu.gt.5 .or. nmenu.lt.0) go to 10
      go to (100,100,200,300,400)nmenu
c
c
c
100   call tgtp1(tgtnam,numtgt,tgtpsi,tgtdia,tgtcat,ntgtpr,tgtval,
     1           ntgts)
      write(6,*)
      write(6,*)'Do you want to change any parameters?(y/n)'
105   read(5,6)yes
      if (yes.ne.'y' .and. yes.ne.'n') go to 105
      if (yes.eq.'y') call parch1(tgtnam,numtgt,tgtpsi,tgtdia,tgtcat,
c                                 ntgtpr,tgtval,ntgts)
      if (yeswgt.eq.'n') then
      write(6,*)'Currently you do not have this program computing'
      write(6,*)'force target values.  Do you want your force target'
      write(6,*)'weights computed for you? (y/n)'
      else
      write(6,*)'Currently you have this program computing force tar-'
      write(6,*)'get values.  Do you still want the program to cal-'
```

```
          write(6,*)'culate them for you? (y/n)'
          endif
110       read(5,6)yeswgt
          if (yeswgt.ne.'y' .and. yeswgt.ne.'n') go to 110
          write(6,*)
          if (yeswgt.eq.'n') go to 5
          call tgtp2(tgtnam,tgtrel,tgtcep,tgtyld,ntgtwh,ntgts,tgtcat)
          write(6,*)'Do you want to change any parameters? (y/n)'
115       read(5,6)yes
          if (yes.ne.'y' .and. yes.ne.'n') go to 115
          if (yes.eq.'y') call parch2(tgtrel,tgtcep,tgtyld,ntgtwh,ntgts,
     c                                tgtcat)
          go to 5
c
c
200       mtgts= 20-ntgts
          write(6,190)
          if (mtgts.le.0) then
          write(6,*)'You can not enter any more target classes unless you'
          write(6,*)'replace an existing target class with a new one.'
          write(6,*)'Do you want to enter any more target classes?(y/n)'
201       read(5,6)yes
          if (yes.ne.'y' .and. yes.ne.'n') go to 201
          if (yes.eq.'y') call tgtexc(ntgts,yeswgt,tgtnam,tgtcat,numtgt,
     c                                tgtpsi,tgtdia,ntgtpr,tgtrel,tgtcep,
     c                                tgtyld,ntgtwh,tgtval)
          go to 5
          endif
205       write(6,*)'You can add ',mtgts,' target classes to your data.'
          write(6,*)'Do you want to add more target classes? (y/n)'
210       read(5,6)yes
          if (yes.ne.'y' .and. yes.ne.'n') go to 210
          if (yes.eq.'n') go to 5
          node=node+1
          ntgts=ntgts+1
          call addtgt(ntgts,yeswgt,tgtnam,tgtcat,numtgt,tgtpsi,
     c              tgtdia,ntgtpr,tgtrel,tgtcep,tgtyld,ntgtwh,
     c              tgtval)
```

```
        go to 200
c
c
300     call tgtexc(ntgts,yeswgt,tgtnam,tgtcat,numtgt,tgtpsi,tgtdia,
     c              ntgtpr,tgtrel,tgtcep,tgtyld,ntgtwh,tgtval)
        node=node+1
        go to 5
c
c
400    write(6,*)'Do you need to see a list of your target class names?'
        write(6,*)'(y/n)'
405     read(5,6)yes
        if (yes.ne.'y' .and. yes.ne.'n') go to 405
        if (yes.eq.'y') call listgt(ntgts,tgtnam)
410     write(6,*)
        write(6,*)'Enter the target class you want to delete. '
        write(6,*)'Type 0 when you are finished.'
412     read(5,7)name
        if (name.eq.'0') go to 5
        node=node+1
        do 415 i=1,ntgts
                if (name.eq.tgtnam(i)) then
                tgtnam(i)=tgtnam(ntgts)
                tgtnam(ntgts)='      '
                tgtcat(i)=tgtcat(ntgts)
                tgtcat(ntgts)=' '
                numtgt(i)=numtgt(ntgts)
                numtgt(ntgts)=0
                tgtpsi(i)=tgtpsi(ntgts)
                tgtpsi(ntgts)='    '
                tgtdia(i)=tgtdia(ntgts)
                tgtdia(ntgts)=0
                ntgtpr(i)=ntgtpr(ntgts)
                ntgtpr(ntgts)=0
                tgtval(i)=tgtval(ntgts)
                tgtval(ntgts)=0
                tgtrel(i)=tgtrel(ntgts)
                tgtrel(ntgts)=0
```

```
                tgtcep(i)=tgtcep(ntgts)
                tgtcep(ntgts)=0
                tgtyld(i)=tgtyld(ntgts)
                tgtyld(ntgts)=0.
                ntgtwh(i)=ntgtwh(ntgts)
                ntgtwh(ntgts)=0
                tgtde(i)=tgtde(ntgts)
                tgtde(ntgts)=0
                ntgts=ntgts-1
                go to 410
                endif
415     continue
        go to 412
190     format(/////////////////////////)
        end
c
c
c
      subroutine wpnedt(nwpns,wpnga,wpname,
     c                  numwpn,numwrh,wpncep,wpnrel,wpnyld,wpndda)
      dimension wpnrel(20),wpncep(20),wpnyld(20),wpndda(20),wpnga(20),
     c          numwrh(20),numwpn(20)
      character yes,name*6,wpname(20)*6
      write(6,*)'The following menu lists possible options to '
      write(6,*)'manipulate the data. (Hit return to continue)'
      read(5,4)yes
    4 format(a1)
    5 write(6,190)
      write(6,*)
      write(6,*)
      write(6,*)'                                MENU'
      write(6,*)
      write(6,*)
      write(6,*)'                          Weapon Class Category'
      write(6,*)
      write(6,*)'     1)      Check/View Data'
      write(6,*)'     2)      Change Parameters'
      write(6,*)'     3)      Add Class'
```

```
      write(6,*)'     4)      Replace Class'
      write(6,*)'     5)      Delete Class'
      write(6,*)
      write(6,*)'Enter your choice.  Type 0 if you do not want any of'
      write(6,*)'these options or are finished with your changes.'
6     format(a1)
7     format(a6)
10    read(5,*)nmenu
      if (nmenu.eq.0) return
      if (nmenu.gt.5 .or. nmenu.lt.0) go to 10
      go to (500,500,600,700,800)nmenu
c
c
500   call wpnp1(nwpns,wpname,numwpn,numwrh,wpnrel,wpncep,wpnyld)
      write(6,*)
      write(6,*)'Do you want any parameters changed? (y/n)'
505   read(5,6)yes
      if (yes.ne.'y' .and. yes.ne.'n') go to 505
      if (yes.eq.'y') call wpnch1(nwpns,wpname,numwpn,numwrh,wpnrel,
     c                            wpnyld,wpncep)
      write(6,*)
      write(6,*)'Do you want to check the weapon alert rates? (y/n)'
510   read(5,6)yes
      if (yes.ne.'y' .and. yes.ne.'n') go to 510
      if (yes.eq.'y') call wpnp2(nwpns,wpname,wpndda,wpnga)
      go to 5
c
c
600   mwpns=20-nwpns
      write(6,190)
      if (mwpns.le.0) then
      write(6,*)'You can not enter any more weapon classes unless you'
      write(6,*)'replace an existing weapon class with a new one.'
      write(6,*)'Do you want to enter any more weapon classes? (y/n)'
601   read(5,6)yes
      if (yes.ne.'y' .and. yes.ne.'n') go to 601
      if (yes.eq.'y') call wpnexc(nwpns,wpname,numwpn,numwrh,wpnrel,
     c                            wpncep,wpnyld,wpndda,wpnga)
```

```
      go to 5
      endif
605   write(6,*)'You can add ',mwpns,' weapon clases to your data.'
      write(6,*)'Are you finished adding weapon classes? (y/n)'
610   read(5,6)yes
      if (yes.ne.'y' .and. yes.ne.'n') go to 610
      if (yes.eq.'y') go to 5
      nwpns=nwpns+1
      call addwpn(nwpns,wpname,numwpn,numwrh,wpnrel,wpncep,wpnyld,
     c                wpndda,wpnga)
      go to 600
c
c
700   call wpnexc(nwpns,wpname,numwpn,numwrh,wpnrel,wpncep,wpnyld,
     c                wpndda,wpnga)
      go to 5
c
c
800   write(6,*)'Do you need to see a list of your weapon class names?'
      write(6,*)'(y/n)'
805   read(5,6)yes
      if (yes.ne.'y' .and. yes.ne.'n') go to 805
      if (yes.eq.'y') call listwp(nwpns,wpname)
      write(6,*)
810   write(6,*)'Enter the weapon class you want to delete.'
      write(6,*)'Type 0 when you are finished.'
812   read(5,7)name
      if (name.eq.'0') go to 5
      do 815 i=1,nwpns
            if (name.eq.wpname(i)) then
            wpname(i)=wpname(nwpns)
            wpname(nwpns)='      '
            numwpn(i)=numwpn(nwpns)
            numwpn(nwpns)=0
            wpnrel(i)=wpnrel(nwpns)
            wpnrel(nwpns)=0
            wpncep(i)=wpncep(nwpns)
            wpncep(nwpns)=0
```

```
                numwrh(i)=numwrh(nwpns)
                numwrh(nwpns)=0
                wpnyld(i)=wpnyld(nwpns)
                wpnyld(nwpns)=0
                wpndda(i)=wpndda(nwpns)
                wpndda(nwpns)=0
                wpnga(i)=wpnga(nwpns)
                wpnga(nwpns)=0
                nwpns=nwpns-1
                go to 810
                endif
815     continue
        go to 812
190     format(////////////////////////)
        end
c
c
c
        subroutine tgtp1(tgtnam,numtgt,tgtpsi,tgtdia,tgtcat,ntgtpr,
     c                   tgtval,ntgts)
        dimension numtgt(20),tgtval(20),tgtdia(20),ntgtpr(20)
        character tgtnam(20)*6,tgtcat(20),tgtpsi(20)*4
        write(6,190)
        write(6,*)'You have entered the following target parameters.'
        write(6,*)
        write(6,*)'TGT NUM  NAME  NUM OF TGTS     VUL       DIAMETER
     c  CAT   PRIORITY VALUE'
        write(6,*)
10      format(3x,i2,5x,a6,2x,i5,9x,a4,5x,f7.2,5x,a1,5x,i1,4x,f7.2)
        do 20 i=1,ntgts
              write(6,10)i,tgtnam(i),numtgt(i),tgtpsi(i),tgtdia(i),
     c                   tgtcat(i),ntgtpr(i),tgtval(i)
20      continue
        write(6,*)
        return
190     format(////////////////////////)
        end
c
```

```
c
      subroutine parch1(tgtnam,numtgt,tgtpsi,tgtdia,tgtcat,ntgtpr,
     c                  tgtval,ntgts)
      dimension numtgt(20),tgtdia(20),ntgtpr(20),tgtval(20)
      character tgtnam(20)*6,tgtcat(20),tgtpsi(20)*4
5     format(a1)
6     format(a4)
7     format(a6)
10    write(6,*)
      write(6,*)'If you have completed your changes type 0, otherwise'
      write(6,*)'type the target class number and hit return.'
      read(5,*)i
      if (i.eq.0) go to 40
      if (i.lt.0 .or. i.gt.ntgts) go to 10
      write(6,*)'Enter the target name and hit return.'
      read(5,7)tgtnam(i)
      write(6,*)'Enter the target category (m,v,f) and hit return.'
30    read(5,5)tgtcat(i)
      if (tgtcat(i).ne.'f' .and. tgtcat(i).ne.'m' .and. tgtcat(i).ne.
     1     'v') go to 30
      write(6,*)'Enter the target VNTK or PSI number (integer) and hit
     1 return'
      read(5,6)tgtpsi(i)
      write(6,*)'Now type in the following data:'
      write(6,*)'number of targets, diameter, priority (1-7), target v
     1alue.'
      read(5,*)numtgt(i),tgtdia(i),ntgtpr(i),tgtval(i)
      go to 10
40    write(6,*)
      return
      end
c
c
c
      subroutine tgtp2(tgtnam,tgtrel,tgtcep,tgtyld,ntgtwh,ntgts,tgtcat)
      dimension tgtrel(20),tgtcep(20),tgtyld(20),ntgtwh(20)
      character tgtnam(20)*6,tgtcat(20)
      write(6,190)
```

```
      write(6,*)'You have these target parameters.'
      write(6,*)
      write(6,*)'  TGT NUM    NAME   RELIABILITY    CEP      YLD
     c  WARHEADS'
50    format(5x,i2,6x,a6,3x,f7.5,4x,f7.3,3x,f6.2,6x,i5)
      do 60 i=1,ntgts
            if (tgtcat(i).eq.'f') write(6,50)i,tgtnam(i),tgtrel(i),
     c      tgtcep(i),tgtyld(i),ntgtwh(i)
60    continue
      write(6,*)
      return
190   format(//'////////////////////////')
      end

c
c
c
      subroutine parch2(tgtrel,tgtcep,tgtyld,ntgtwh,ntgts,tgtcat)
      dimension tgtrel(20),tgtcep(20),tgtyld(20),ntgtwh(20)
      character tgtcat(25)
10    write(6,*)'Enter the target class number to be changed.'
      write(6,*)'Enter 0 if you are finished with your changes.'
      read(5,*)i
      if (i.eq.0) return
      if (i.lt.0 .or. i.gt.ntgts) go to 10
      if (tgtcat(i).eq.'f') then
          write(6,*)'Enter in the following data:'
      write(6,*)'reliability, CEP (NM), yield (MT), warheads per targe
     1t.'
          read(5,*) tgtrel(i),tgtcep(i),tgtyld(i),ntgtwh(i)
          write(6,*)
      else
          write(6,*)'The category of this target class is not a '
          write(6,*)'force type so the data does not need to be '
          write(6,*)'changed.'
          write(6,*)
      endif
      go to 10
      end
```

```
c
c
      subroutine listgt(ntgts,tgtnam)
      character tgtnam(20)*6
      write(6,20)
      write(6,*)'          TARGET CLASS NAME'
      write(6,*)
15    format(12x,a6)
      do 10 i=1,ntgts
10    write(6,15)tgtnam(i)
      return
20    format(/////////////////////)
      end
c
c
c
      subroutine tgtexc(ntgts,yeswgt,tgtnam,tgtcat,numtgt,tgtpsi,
     c                  tgtdia,ntgtpr,tgtrel,tgtcep,tgtyld,ntgtwh,
     c                  tgtval)
      dimension numtgt(20),tgtdia(20),ntgtpr(20),tgtval(20),
     c          tgtrel(20),tgtcep(20),tgtyld(20),ntgtwh(20)
      character yeswgt,tgtnam(20)*6,tgtcat(20),yes,name*6,tgtpsi(20)*4
5     format(a1)
6     format(a6)
      write(6,*)'Do you want a list of the current target classes?'
      write(6,*)'(y/n)'
10    read(5,5)yes
      if (yes.ne.'y' .and. yes.ne.'n') go to 10
      if (yes.eq.'y') call listgt(ntgts,tgtnam)
      write(6,*)
20    write(6,*)'Type 0 if you are finished, otherwise enter a'
      write(6,*)'target class name you want to replace.'
25    read(5,6)name
      if (name.eq '0')return
      do 30 i=1,ntgts
            if (name.eq.tgtnam(i)) then
                  n=i
                  go to 40
```

```
            endif
30      continue
        go to 25
40      call addtgt(n,yeswgt,tgtnam,tgtcat,numtgt,tgtpsi,tgtdia,ntgtpr,
     c                tgtrel,tgtcep,tgtyld,ntgtwh,tgtval)
        write(6,*)
        go to 20
        end
c
c
c
        subroutine addtgt(n,yeswgt,tgtnam,tgtcat,numtgt,tgtpsi,tgtdia,
     c                    ntgtpr,tgtrel,tgtcep,tgtyld,ntgtwh,tgtval)
        dimension numtgt(20),tgtdia(20),ntgtpr(20),tgtrel(20),
     c            tgtcep(20),tgtyld(20),ntgtwh(20),tgtval(20)
        character yeswgt,tgtnam(20)*6,tgtcat(20),tgtpsi(20)*4
        write(6,*)
5       format(a1)
6       format(a4)
7       format(a6)
        write(6,*)'Enter your new target class name.'
        read(5,7)tgtnam(n)
10      write(6,*)'Enter the category (v,m,f) for your target class.'
        read(5,5)tgtcat(n)
        if (tgtcat(n).ne.'f' .and. tgtcat(n).ne.'m' .and. tgtcat(n).ne.
     c  'v') go to 10
        write(6,*)'Enter the target VNTK or PSI (integer) number.'
        read(5,6)tgtpsi(n)
        write(6,*)'Enter the following data:'
        if (yeswgt.eq.'y' .and. tgtcat(n).eq.'f') then
          write(6,*)'number of targets, diameter (NM), priority(1-7), re
     lliability,'
          write(6,*)'CEP (NM), yield (MT), warheads per tgt.'
          read(5,*)numtgt(n),tgtdia(n),ntgtpr(n),
     c    tgtrel(n),tgtcep(n),tgtyld(n),ntgtwh(n)
        else
          write(6,*)'Number of targets, diameter (NM), priority (1-7), v
     lalue.'
```

```
          read(5,*)numtgt(n),tgtdia(n),ntgtpr(n),
     c    tgtval(n)
        endif
        return
        end
c
c
c
        subroutine wpnp1(nwpns,wpname,numwpn,numwrh,wpnrel,wpncep,
     c                   wpnyld)
        dimension numwpn(20),numwrh(20),wpnrel(20),wpncep(20),wpnyld(20)
        character wpname(20)*6
        write(6,190)
        write(6,*)'You have entered the following target parameters:'
        write(6,*)
       write(6,*)'WPN NUM  NAME  NUM OF WPNS  WARHDS/WPN RELIABILITY
     cCEP      YLD'
        write(6,*)
10      format(3x,i2,5x,a6,3x,i5,7x,i3,6x,f6.2,5x,f7.3,3x,f7.2)
        do 20 i=1,nwpns
              write(6,10)i,wpname(i),numwpn(i),numwrh(i),wpnrel(i),
     c                   wpncep(i),wpnyld(i)
20      continue
        write(6,*)
        return
190     format(//////////////////////)
        end
c
c
c
        subroutine wpnch1(nwpns,wpname,numwpn,numwrh,wpnrel,wpnyld,
     c                  wpncep)
        dimension numwpn(20),numwrh(20),wpnrel(20),wpnyld(20),wpncep(20)
        character wpname(20)*6
5       format(a6)
10      write(6,*)
        write(6,*)
        write(6,*)'Type the weapon class number and hit return.'
```

```
      write(6,*)'If you have completed your changes type 0.'
15    read(5,*)i
      if (i.eq.0) go to 40
      if (i.lt.0 .or. i.gt.nwpns) go to 15
      write(6,*)'Type in the weapon name and hit return.'
      read(5,5)wpname(i)
      write(6,*)
      write(6,*)
      write(6,*)'Type in thw following data:'
      write(6,*)'number of weapons, warheads per weapon, reliability,
     c CEP (NM), yield (MT).'
      read(5,*) numwpn(i),numwrh(i),wpnrel(i),wpncep(i),wpnyld(i)
      go to 10
40    write(6,*)
      return
      end
c
c
c
      subroutine wpnp2(nwpns,wpname,wpndda,wpnga)
      dimension wpndda(20),wpnga(20)
      character wpname(20)*6,yes
 5    write(6,190)
      write(6,*)' You have entered the following alert rates'
      write(6,*)
      write(6,*)' WPN NUM    NAME     DAY-TO DAY    GENERATED'
      write(6,*)
10    format(3x,i2,6x,a6,5x,f6.2,8x,f6.2)
      do 20 i=1,nwpns
20          write(6,10)i,wpname(i),wpndda(i),wpnga(i)
      write(6,*)
      write(6,*)'Is the data O.K.? y/n'
12    read(5,15)yes
      if(yes.ne.'y' .and. yes.ne.'n')go to 12
      if(yes.eq.'y') return
15    format(a1)
      call wpnch2(nwpns,wpndda,wpnga)
      go to 5
```

```
190     format(////////////////////////)
        end
c
c
c
        subroutine wpnch2(nwpns,wpndda,wpnga)
        dimension wpndda(20),wpnga(20)
10      write(6,*)
        write(6,*)'Type 0 if finished changing alert rates, otherwise'
        write(6,*)'input the weapon class number to be changed.'
11      read(5,*)i
        if (i.eq.0) go to 40
        if (i.lt.0 .or. i.gt.nwpns) go to 11
        write(6,*)'Type in the following data:'
        write(6,*)'daily alert rate, generated alert rate.'
        read(5,*)wpndda(i),wpnga(i)
        go to 10
40      write(6,*)
        return
        end
c
c
c
        subroutine wpnexc(nwpns,wpname,numwpn,numwrh,wpnrel,wpncep,
     c                    wpnyld,wpndda,wpnga)
        dimension numwpn(20),numwrh(20),wpncep(20),wpnyld(20),
     c            wpnrel(20),wpndda(20),wpnga(20)
        character wpname(20)*6,yes,name*6
5       format(a1)
        write(6,*)'Do you want a list of current weapon class names?'
        write(6,*)'(y/n)'
10      read(5,5)yes
        if (yes.ne.'y' .and. yes.ne.'n') go to 10
        if (yes.eq.'y') call listwp(nwpns,wpname)
        write(6,*)
12      write(6,*)'Enter the weapon class name you want to replace.'
        write(6,*)'Type 0 if you are finished:'
14      read(5,15)name
```

```
15      format(a6)
        if (name.eq.'0') return
        do 20 i=1,nwpns
                if (wpname(i).eq.name) then
                        n=i
                        go to 30
                endif
20      continue
        go to 14
30      call addwpn(n,wpname,numwpn,numwrh,wpnrel,wpncep,wpnyld,wpndda,
     c                wpnga)
        go to 12
        end
c
c
c
        subroutine addwpn(n,wpname,numwpn,numwrh,wpnrel,wpncep,wpnyld,
     c                    wpndda,wpnga)
      dimension numwpn(20),numwrh(20),wpnrel(20),wpncep(20),wpnyld(20),
     c           wpndda(20),wpnga(20)
        character wpname(20)*6
        write(6,*)
        write(6,*)'Enter your new weapon class name.'
5       format(a6)
        read(5,5)wpname(n)
        write(6,*)
        write(6,*)'Enter the following data:'
        write(6,*)'number of weapons, warheads per weapon, reliability,
     c CEP, yield,'
        write(6,*)'daily alert rate, generated alert rate.'
        read(5,*)numwpn(n),numwrh(n),wpnrel(n),wpncep(n),wpnyld(n),
     c           wpndda(n),wpnga(n)
        write(6,*)
        return
        end
c
c
        subroutine listwp(nwpns,wpname)
```

```
      character wpname(20)*6
      write(6,190)
      write(6,*)'       WEAPON CLASS NAME'
      write(6,*)
5     format(15x,a6)
      do 10 i=1,nwpns
      write(6,5)wpname(i)
10    continue
      return
190   format(/////////////////////)
      end


      subroutine object(tgtde)
      common/main/ ntof(10),iprin(61),isub(10,20),itype(10,20),
     1 wght(10,20),wpnyld(20),tgtval(20),ntgtpr(20),rhs(61),nprit,
     1 nhedg,numtgt(20),wpncep(20)
      common/aijin1/ nwpns,ntgts,sparse(20,20),aij(61,401),icount
      common/cat/tgtcat(20)
      common/bnd/ibnd
      dimension tgtde(20)
      character tgtcat(20),yes
      ibnd=0
      write(6,310)
      write(6,*)'This section builds your objective function.  The '
      write(6,*)'function is divided into 3 major categories.  For '
      write(6,*)'complete explanation of your options please consult'
      write(6,*)'the user guide.  Please select one of the following.'
      write(6,*)'Enter 1, 2, or 3.'
      write(6,*)'     1) This problem requires allocation restricted'
      write(6,*)'        to the available arsenal.'
      write(6,*)'     2) This problem requires all goals to be met'
      write(6,*)'        regardless of the available forces.'
      write(6,*)'     3) This problem requires allocation restricted'
      write(6,*)'        to the available arsenal. Also, the DE goals'
      write(6,*)'        are converted to upper bounds.  If this pro-'
      write(6,*)'        blem is picked, the hedging options must be '
      write(6,*)'        used to drive the allocation.'
```

```
  100   read(5,*) ntemp
        if(ntemp.lt.1 .or. ntemp .gt.3) go to 100
        mtemp=nwpns+ntgts
        do 105 mn=1,mtemp
           aij(mn,1)=rhs(mn)
  105   continue
c
c       Problem restricted to available arsenal.
c
        if(ntemp.eq.1) then
c
c       Objective function for P1.
c
          do 110 i=1,nwpns
                isub(1,i)=ntgts+i
                itype(1,i)=3
                wght(1,i)=1.0
                iprin(ntgts+i)=1
  110     continue
          ntof(1)=nwpns
c
c       Objective function for targeting goals.
c
          max=0
          do 120 i=1,ntgts
                itemp=ntgtpr(i) + 2
                if(itemp.gt.9) itemp=9
                if(itemp.gt. max) max=itemp
                ntof(itemp)=ntof(itemp)+1
                nprior=ntof(itemp)
                isub(itemp,nprior)=i
                itype(itemp,nprior)=4
                wght(itemp,nprior)=tgtval(i)*(1-tgtde(i))
                iprin(i)=itemp
  120     continue
c
c       Objective term for extreme goal.
c
```

```
      nprit=max+1
      isub(nprit,1)=ntgts+nwpns+1
      itype(nprit,1)=3
      wght(nprit,1)=1.0
      icon=ntgts+nwpns+1
      aij(icon,i)=0.0
      iprin(icon)=nprit
      ntof(nprit)=1
    write(6,310)
    write(6,*)'You have the opportunity to select a lowest priority'
    write(6,*)'goal.  Please select one of the following.'
    write(6,*)'If there are any weapons remaining following the'
    write(6,*)'allocation, this priority will be used.  If any of'
    write(6,*)'your goals are not met, it will not be used.'
    write(6,*)'Enter 1, 2, 3, 4.'
    write(6,*)'   1)  Minimize warheads used.'
    write(6,*)'   2)  Minimize megatonnage used.'
    write(6,*)'   3)  Minimize countermilitary potential.'
    write(6,*)'   4)  Minimize total equivalent megatonnage.'
130   read(5,*) m
      if(m.lt.1 .or. m.gt.4) go to 130
      if(m.eq.1) then
        ntvar=nwpns*ntgts
          do 140 i=1,ntvar
            aij(icon,(i+1))=1.0
140       continue
      endif
      if(m.eq.2) then
          do 160 i=1,nwpns
            tyld=wpnyld(i)
            do 150 j=1,ntgts
              aij(icon,(1+j+(i-1)*ntgts))=tyld
150         continue
160       continue
      endif
      if(m.eq.3) then
          do 180 i=1,nwpns
            if(wpnyld(i).lt.0.2) then
```

```
                        tcmp=wpnyld(i)**(0.8)/(wpncep(i)*6)**2
                    else
                        tcmp=wpnyld(i)**(0.6666)/(wpncep(i)*6)**2
                    endif
                    if(tcmp.gt.3) tcmp=3
                    do 170 j=1,ntgts
                        aij(icon,(1+j+(i-1)*ntgts))=tcmp
  170               continue
  180             continue
                endif
                if(m.eq.4) then
                    do 200 i=1,nwpns
                        if(wpnyld(i).lt.1.0) then
                            emt=wpnyld(i)**0.6666
                        else
                            emt=wpnyld(i)**0.5
                        endif
                        do 190 j=1,ntgts
                            aij(icon,(1+j+(i-1)*ntgts))=emt
  190               continue
  200             continue
                endif
            endif
c
c           Type 2 objective function.
c
            if(ntemp.eq.2) then
c
c           Bounding by DE goals, also insure goals are met
c
              do 210 j=1,ntgts
                    isub(1,j)=j
                    itype(1,j)=3
                    wght(1,j)=1.0
                    iprin(j)=1
                    isub(3,j)=j
                    itype(3,j)=4
                    wght(3,j)=1.0
```

```
  210     continue
          ntof(1)=ntgts
          ntof(3)=ntgts
c
c       Insure available weapons are used prior to building more.
c
          do 220 i=1,nwpns
                isub(4,i)=ntgts+i
                itype(4,i)=4
                wght(4,i)=1.0
                iprin(ntgts+i)=4
  220     continue
          ntof(4)=nwpns
        write(6,310)
        write(6,*)'Do you wish to set all available forces to zero?'
        write(6,*)'If yes, I will assign only the best wpns to the'
        write(6,*)'targets.  If no, I will use all of the existing'
        write(6,*)'inventory prior to selecting more weapons. (y/n)'
  230     read(5,231) yes
  231     format(a1)
          if(yes.ne.'y' .and. yes.ne.'n') go to 230
          if(yes.eq.'y') then
             do 240 mn=1,nwpns
                aij((ntgts+mn),1)=0.0
  240        continue
          endif
c
c       The following is the extreme goal, insuring a minimum aspect of the'
c       arsenal built.
c
          icon=ntgts+nwpns+1
          isub(5,1)=icon
          itype(5,1)=3
          wght(5,1)=1
          aij(icon,1)=0
          iprin(icon)=5
          ntof(5)=1
        write(6,310)
```

```
      write(6,*)'The arsenal to be built should have one of the'
      write(6,*)'following characteristics.  Please select one. '
      write(6,*)'Input 1, 2, 3, or 4.'
      write(6,*)'   1)  Minimize warheads used.'
      write(6,*)'   2)  Minimize megatonnage used.'
      write(6,*)'   3)  Minimize countermilitary potential.'
      write(6,*)'   4)  Minimize total equivalent megatonnage.'
241      read(5,*) m
         if(m.lt.1 .or. m.gt.4) go to 241
         if(m.eq.1) then
           ntvar=nwpns*ntgts
               do 242 i=1,ntvar
                 aij(icon,(i+1))=1.0
242            continue
         endif
         if(m.eq.2) then
           do 260 i=1,nwpns
               tyld=wpnyld(i)
               do 250 j=1,ntgts
                 aij(icon,(1+j+(i-1)*ntgts))=tyld
250            continue
260        continue
         endif
         if(m.eq.3) then
           do 280 i=1,nwpns
               if(wpnyld(i).lt.0.2) then
                  tcmp=wpnyld(i)**(0.8)/(wpncep(i)*6)**2
               else
                  tcmp=wpnyld(i)**(0.6666)/(wpncep(i)*6)**2
               endif
               if(tcmp.gt.3) tcmp=3
               do 270 j=1,ntgts
                  aij(icon,(1+j+(i-1)*ntgts))=tcmp
270            continue
280        continue
         endif
         if(m.eq.4) then
           do 300 i=1,nwpns
```

```
                  if(wpnyld(i).lt.1.0) then
                     emt=wpnyld(i)**0.6666
                  else
                     emt=wpnyld(i)**0.5
                  endif
                  do 290 j=1,ntgts
                     aij(icon,(1+j+(i-1)*ntgts))=emt
  290             continue
  300          continue
            endif
            nprit=5
            endif
            if(ntemp.eq.3) then
c
c           Objective function for P1, bounding with weapon availability.
c
            do 305 i=1,nwpns
                  isub(1,i)=ntgts+i
                  itype(1,i)=3
                  wght(1,i)=1.0
                  iprin(ntgts+i)=1
 305        continue
            ntof(1)=nwpns
c
c           Objective function for P2, bounding with targeting goals.
c
            do 320 j=1,ntgts
                  isub(2,j)=j
                  itype(2,j)=3
                  wght(2,j)=1.0
                  iprin(j)=2
 320        continue
            ntof(2)=ntgts
            ibnd=1
c
c           The following is the extreme goal.  It is the lowest priority
c           goal.
c
```

```
      icon=ntgts+nwpns+1
      isub(4,1)=icon
      itype(4,1)=3
      wght(4,1)=1
      aij(icon,1)=0.0
      iprin(icon)=4
      ntof(4)=1
      write(6,310)
      write(6,*)'The following is the lowest priority goal for this'
      write(6,*)'allocation.  Please select one.  Input 1, 2, 3, 4,'
      write(6,*)'or 5.'
      write(6,*)'   1)  Minimize warheads used.'
      write(6,*)'   2)  Minimize megatonnage used.'
      write(6,*)'   3)  Minimize countermilitary potential.'
      write(6,*)'   4)  Minimize total equivalent megatonnage.'
      write(6,*)'   5)  Use as much of the remaining arsenal as '
      write(6,*)'       possible.'
341   read(5,*)m
        if(m.lt.1 .or. m.gt.5) go to 341
        if(m.eq.1) then
          ntvar=nwpns*ntgts
              do 342 i=1,ntvar
               aij(icon,(i+1))=1.0
342           continue
        endif
        if(m.eq.2) then
          do 360 i=1,nwpns
              tyld=wpnyld(i)
              do 350 j=1,ntgts
                aij(icon,(1+j+(i-1)*ntgts))=tyld
350           continue
360       continue
        endif
        if(m.eq.3) then
          do 380 i=1,nwpns
              if(wpnyld(i).lt.0.2) then
                tcmp=wpnyld(i)**(0.8)/(wpncep(i)*6)**2
              else
```

```
                tcmp=wpnyld(i)**(0.6666)/(wpncep(i)*6)**2
              endif
              if(tcmp.gt.3) tcmp=3
              do 370 j=1,ntgts
                aij(icon,(1+j+(i-1)*ntgts))=tcmp
370           continue
380         continue
          endif
          if(m.eq.4) then
            do 400 i=1,nwpns
              if(wpnyld(i).lt.1.0) then
                emt=wpnyld(i)**0.6666
              else
                emt=wpnyld(i)**0.5
              endif
              do 390 j=1,ntgts
                aij(icon,(1+j+(i-1)*ntgts))=emt
390           continue
400         continue
          endif
          if(m.eq.5) then
          write(6,310)
          itemp=0
          do 420 i=1,nwpns
               itemp=itemp+aij(ntgts+i,1)
420       continue
          aij(icon,1)=itemp
          itype(4,1)=4
          wght(4,1)=1
          write(6,*)'This option gives you the capability to use the'
          write(6,*)'remainder of your weapons.  They will be allo-'
          write(6,*)'cated according to one of the following options.'
          write(6,*)'Regardless of the number of weapons available,'
          write(6,*)'your DE bounds will not be exceeded.  Dump weapons'
          write(6,*)'on '
          write(6,*)'   1)  Force targets.'
          write(6,*)'   2)  Other Military targets.'
          write(6,*)'   3)  Value targets.'
```

```
      write(6,*)'   4)  Any remaining targets.'
510   read(5,*)m
      if(m.lt.1 .or. m.gt.4) go to 510
      if(m.eq.1) then
            do 525 j=1,ntgts
              if(tgtcat(j).eq.'f') then
               do 520 i=1,nwpns
                aij(icon,(1+j+(i-1)*ntgts))=1
520             continue
              endif
525         continue
      endif
      if(m.eq.2) then
            do 535 j=1,ntgts
              if(tgtcat(j).eq.'m') then
               do 530 i=1,nwpns
                aij(icon,(1+j+(i-1)*ntgts))=1
530             continue
              endif
535         continue
      endif
      if(m.eq.3) then
            do 545 j=1,ntgts
              if(tgtcat(j).eq.'v') then
               do 540 i=1,nwpns
                aij(icon,(1+j+(i-1)*ntgts))=1
540             continue
              endif
545         continue
      endif
      if(m.eq.4) then
            do 555 j=1,ntgts
               do 550 i=1,nwpns
                aij(icon,(1+j+(i-1)*ntgts))=1
550             continue
555         continue
      endif
     endif
```

```
          nprit=4
          endif
  310     format(////////////////////)
          return
          end
c
c
c
          subroutine wtintr
          common/aijin1/ nwpns,ntgts,sparse(20,20),aij(61,401),icount
          common/wtint/ wpname(20),tgtnam(20)
          character wpname(20)*6,tgtnam(20)*6,ab
1         format(a1)
          write(6,600)
          write(6,*)'This routine designates which weapon/target assign-'
          write(6,*)'ments are inappropriate.  It is used when there is'
          write(6,*)'some reason the analyst does not want a designated'
          write(6,*)'weapon class to be assigned to a designated target'
          write(6,*)'class.  There is no limit to the number of designa-'
          write(6,*)'tions made.  (Hit return to continue)'
          read(5,1)ab
          if(ntgts .ge. nwpns) ntemp=ntgts
          if(nwpns .gt. ntgts) ntemp=nwpns
          write(6,*)
          write(6,*)' NO.  WEAPONS          NO. TARGETS'
          write(6,*)
          do 1020 i=1,ntemp
          write(6,1030) i,wpname(i),i,tgtnam(i)
 1020     continue
 1030     format(1x,i3,4x,a6,t23,i3,4x,a6)
  100     write(6,*)
          write(6,*)'Input an ordered pair of weapon number and target '
          write(6,*)'number.  An entry of 0,1 will get you out of this'
          write(6,*)'section.'
  105     read(5,*) i,j
          if(i.lt.0.or.i.gt.nwpns.or.j.lt.1.or.j.gt.ntgts) go to 105
          if(i.eq.0) go to 500
          sparse(j,i)=1.0
```

```
      go to 100
  500 return
  600 format(//////////////////////)
      end


      subroutine hedge
      common/main/ ntof(10),iprin(61),isub(10,20),itype(10,20),
     1 wght(10,20),wpnyld(20),tgtval(20),ntgtpr(20),rhs(61),nprit,
     1 nhedg,numtgt(20),wpncep(20)
      common/aijin1/ nwpns,ntgts,sparse(20,20),aij(61,401),icount
      common/wtint/ wpname(20),tgtnam(20)
      common/bnd/ibnd
      character wpname(20)*6,tgtnam(20)*6,ab
      integer hedglt
    1 format(a1)
      nhedg=0
      nvar=0
      itsub=ntgts+nwpns+1
      if=2
      if(ibnd.eq.1)if=3
      write(6,*)
      write(6,*)'This section permits selection of user defined goals.'
      write(6,*)'There are seven types to choose from.  Select any'
      write(6,*)'particular type by inputting the appropriate number.'
      write(6,*)'For further information about hedging, see the users'
      write(6,*)'guide.'
 1000 format(' You have space to select',i3,' hedging constraints.')
 1010 hedglt= 20 - nvar
      if (hedglt.le.0) then
               inum=0
               go to 1016
               endif
      write(6,1000) hedglt
      write(6,*)'The following is a list of hedging types available.'
      write(6,*)'Type 0 to exit this section.'
      write(6,*)'  1) Enforce a minimum level of damage on a particu-'
      write(6,*)'     lar target class.'
```

```
      write(6,*)'  2) Enforce a minimum level of damage on a particu-'
      write(6,*)'     lar target class using a specific set of weapons.
     1 '
      write(6,*)'  3) Enforce an upper level of damage on a particu-'
      write(6,*)'     lar target class resulting from a specific set'
      write(6,*)'     of weapons.'
      write(6,*)'  4) Restrict the number of a class of weapons which'
      write(6,*)'     can be allocated to a target class.'
      write(6,*)'  5) Build your own constraint.'
      write(6,*)'  6) Enforce a minimum level of damage on a particu-'
      write(6,*)'     lar set of target classes.'
      write(6,*)'  7) Restrict the number of weapons which can be al-'
      write(6,*)'     located to each target in a particular target'
      write(6,*)'     class.'
 1015  read(5,*) inum
 1016  if (inum .eq. 0) then
           write(6,3100)
           write(6,1017) nhedg
 1017      format('You have added',i4,' hedges.')
           write(6,*)'Is this the right number? y/n'
200 1018   read(5,1)ab
           if(ab.ne.'y' .and. ab.ne.'n') go to 1018
           if(ab.eq.'n') go to 1010
           write(6,3100)
           write(6,*)'                                        PLEASE STANDBY'
           write(6,3105)
           ntof(if)=nvar
           go to 3050
       endif
       if(inum.lt.1 .or. inum.gt.7) go to 1015
       if(ntgts .ge. nwpns)ntemp=ntgts
       if(nwpns .gt. ntgts)ntemp=nwpns
       write(6,*)
       write(6,*)'NO.  WEAPONS    AMT.   NO.  TARGETS  AMT.'
       write(6,*)
       do 1020 i=1,ntemp
       write(6,1030) i,wpname(i),rhs(ntgts+i),i,tgtnam(i),numtgt(i)
 1020  continue
```

```
 1030   format(1x,i2,4x,a6,2x,f6.1,t25,i2,4x,a6,2x,i4)
        nhedg=nhedg+1
        nvar=nvar+1
        itsub=itsub+1
        iprin(itsub)=if
        isub(if,nvar)=itsub
        wght(if,nvar)=1.0
        if(inum .eq. 1)then
           itype(if,nvar)=4
           go to 2010
        endif
        if(inum .eq. 2)then
           itype(if,nvar)=4
           go to 2040
        endif
        if(inum .eq. 3)then
           itype(if,nvar)=3
           go to 2040
        endif
        if(inum .eq. 6) go to 2999
c
c       Type 7
c
        if(inum.eq.7) then
          itype(if,nvar)=3
          write(6,*)
          write(6,*)'Input the target class number and the maximum'
          write(6,*)'number of warheads to be allocated to each target.'
 1032     read(5,*)j,t
          if(j.lt.1 .or. j.gt.ntgts .or. t.lt.0.0) go to 1032
          do 1035 i=1,nwpns
 1035      aij(itsub,(1+j+((i-1)*ntgts)))=1.0
          aij(itsub,1)=t*numtgt(j)
          write(6,3105)
          go to 1010
        endif
c
c       Type 4
```

201

```
c
        if(inum .eq. 4)then
           itype(if,nvar)=3
        write(6,*)
        write(6,*)'Input weapon number, target number, and'
        write(6,*)'maximum number of warheads.'
 1040      read(5,*) i,j,trhs
           if(j.gt.ntgts.or.i.gt.nwpns.or.trhs.lt.0.0.or.j.lt.1.or.
     1     i.lt.1) go to 1040
           aij(itsub,(1+j+((i-1)*ntgts)))=1.0
           aij(itsub,1)=trhs
           write(6,3100)
           go to 1010
        endif
c
c       Type 5
c
        if(inum .eq. 5)then
        write(6,3100)
        write(6,*)'This type of constraint is intended for the advanced'
        write(6,*)'user.  Before proceeding, please consult the user'
        write(6,*)'guide.  At any time, an input of 0 will return you to'
        write(6,*)'the basic hedging menu.'
        write(6,*)'Input the RHS value. (it must be positive)'
 1050      read(5,*) trhs
           if(trhs.lt.-0.01) go to 1050
           If(trhs .gt. -0.001 .and. trhs .lt. 0.001) then
             nhedg=nhedg-1
             itsub=itsub-1
             nvar=nvar-1
             write(6,3100)
             go to 1010
           endif
           write(6,*)
           write(6,*)'What type is this constraint? Select 1, 2, or 3.'
           write(6,*)'     1) LE.'
           write(6,*)'     2) GE.'
           write(6,*)'     3) EQ.'
```

```
1060      read(5,*)itemp
          if(itemp.lt.0 .or. itemp.gt.3) go to 1060
          if(itemp .eq. 0) then
            nhedg=nhedg-1
            itsub=itsub-1
            nvar=nvar-1
            write(6,3100)
            go to 1010
          endif
          if(itemp.eq.1) itype(if,nvar)=3
          if(itemp.eq.2) itype(if,nvar)=4
          if(itemp.eq.3) then
            itype(if,nvar)=3
            nvar=nvar+1
            isub(if,nvar)=itsub
            wght(if,nvar)=1
            itype(if,nvar)=4
          endif
          write(6,3100)
      write(6,*)'This section adds your variable coefficients.'
      write(6,*)'You will be asked to input weapon type i, target '
      write(6,*)'class j, and the coefficient.  The first two numbers'
      write(6,*)'must be integers and the third can be real.  For'
      write(6,*)'example, type 4, 2, 1.3 yields a coefficient of 1.3'
      write(6,*)'on variable Y42.  Hit return to continue.'
       read(5,1)ab
          write(6,*)
          write(6,*)'NO.  WEAPONS    AMT.    NO.  TARGETS  AMT.'
          write(6,*)
          do 1080 i=1,ntemp
          write(6,1030) i,wpname(i),rhs(ntgts+i),i,tgtnam(i),numtgt(i)
1080      continue
      write(6,*)
       write(6,*)
       write(6,*)'Input weapon number, target number, and coefficient.'
       write(6,*)'0,1,1 will exit this section without a constraint.'
1090      read(5,*) i,j,coef
          if(i.lt.0.or.j.lt.1.or.i.gt.nwpns.or.j.gt.ntgts)go to 1090
```

```
                if(i.eq.0)then
      c
      c         Exit without a constraint
      c
                     nvar=nvar-1
                     itsub=itsub-1
                     nhedg=nhedg-1
                     if(itemp.eq.3)nvar=nvar-1
                     write(6,3100)
                     go to 1010
                endif
                aij(itsub,(1+j+(i-1)*ntgts))=coef
       2000     write(6,*)'Input warhead type i, target class j, and the '
                write(6,*)'coefficient.  0,1,1 will exit this section.'
       2005     read(5,*) i,j,coef
                if(i.lt.0.or.j.lt.1.or.i.gt.nwpns.or.j.gt.ntgts)go to 2005
                if(i.eq.0) then
      c
      c         Exit with a constraint
      c
                     aij(itsub,1)=trhs
                     write(6,3100)
                     go to 1010
                endif
                aij(itsub,(1+j+(i-1)*ntgts))=coef
                go to 2000
                endif
       2010     continue
      c
      c         Hedging type 1
      c
                write(6,*)
                write(6,*)'Input the target class number.'
       2020     read(5,*)inum
                if(inum.lt.1 .or. inum.gt.ntgts) go to 2020
                write(6,*)'Input your desired damage expectancy level.'
                write(6,*)'It must be a real number between 0 and 1.'
       2030     read(5,*)temp
```

```
            if(temp.le.0.0 .or. temp.ge.1.0) go to 2030
            aij(itsub,1)=-1*numtgt(inum)*alog(1-temp)
c
c       Target class number equals inum
c       Use sparse coefficient to put ssps into constraint
c       itsub in Aij
c
            do 2035 i=1,nwpns
                 aij(itsub,(1+inum+(i-1)*ntgts))=-1*alog(sparse(inum,i))
 2035       continue
            write(6,3100)
            go to 1010
c
c       Given a type 2 or 3 constraint, here is where the variables
c       and RHS are input.
c
 2040       write(6,*)'Input the target class number.'
 2041       read(5,*) inum
            if (inum.lt. 1 .or. inum.gt.ntgts) go to 2041
            write(6,*)'Input your desired damage expectancy level.'
            write(6,*)'It must be a real number between 0 and 1.'
 2050       read(5,*) temp
            if(temp.le.0.0 .or. temp.ge.1.0) go to 2050
          write(6,*)'Input a weapon number to begin building your'
          write(6,*)'specific set.  A zero will cause you to exit this'
          write(6,*)'section without a constraint.'
 2060       read(5,*) i
            if(i.lt.0 .or. i.gt.nwpns) go to 2060
            if(i.eq.0) then
                 nhedg=nhedg-1
                 itsub=itsub-1
                 nvar=nvar-1
                 write(6,3100)
                 go to 1010
            endif
            aij(itsub,1)=-1*numtgt(inum)*alog(1-temp)
            aij(itsub,(1+inum+(i-1)*ntgts))=-1*alog(sparse(inum,i))
 2070       write(6,*)'Input another weapon number for your specific'
```

```
            write(6,*)'set.  0 will exit this section.'
 2075       read(5,*) i
            if(i.lt.0 .or. i.gt. nwpns) go to 2075
            if(i.eq.0) then
                 write(6,3100)
                 go to 1010
            endif
            aij(itsub,(1+inum+(i-1)*ntgts))=-1*alog(sparse(inum,i))
            go to 2070
c
c       Case 6
c
 2999       itype(if,nvar)=4
            ntot=0
            write(6,*)
          write(6,*)'Input your desired damage expectancy level.  It must'
          write(6,*)'be a real between 0 and 1.'
 3000       read(5,*) temp
            if(temp.le.0.0 .or. temp.ge.1.0) go to 3000
        write(6,*)'Input a target class number to begin building'
        write(6,*)'your specific set.  A zero will cause you to exit'
        write(6,*)'this section without a new constraint. '
 3010       read(5,*) j
            if(j.lt.0 .or. j.gt.ntgts) go to 3010
            if (j.eq.0) then
                 nhedg=nhedg-1
                 itsub=itsub-1
                 nvar=nvar-1
                 write(6,3100)
                 go to 1010
            endif
            ntot=ntot+numtgt(j)
            do 3020 i=1,nwpns
              aij(itsub,(1+j+(i-1)*ntgts))=-1*alog(sparse(j,i))
 3020       continue
 3030     write(6,*)'Input another target number for your specific set.'
          write(6,*)'Zero will exit this section.'
 3035       read(5,*) j
```

```
            if(j.lt.0 .or. j.gt.ntgts) go to 3035
            if(j.eq.0) then
                 aij(itsub,1)=-1*ntot*alog(1-temp)
                 write(6,*)
                 go to 1010
            endif
            do 3040 i=1,nwpns
                 aij(itsub,(1+j+(i-1)*ntgts))=-1*alog(sparse(j,i))
 3040       continue
            ntot=ntot+numtgt(j)
            go to 3030
 3050       return
 3100       format(////////////////////)
 3105       format(//////////)
 3150       format(a1)
       end
c
c
c
       subroutine aijin
       common/aijin1/ nwpns,ntgts,sparse(20,20),aij(61,401),icount
c
c      This subroutine takes the number of targets, the number of weapons,
c      and the matrix Sparse, and builds the first ntgts+nwpns rows of
c      matrix Aij.
c
c      The following takes the values from sparse and puts their negative
c      natural logs into aij.
c
       do 200 i=1,nwpns
         do 210 j=1,ntgts
           aij(j,(1+j+(i-1)*ntgts))=-1*alog(sparse(j,i))
 210     continue
 200   continue
c
c      The following puts a 1 into the appropriate spots in the weapons
c      rows.
c
```

```
      do 320 i=1,nwpns
         itemp=ntgts+i
         do 300 j=1,ntgts
           aij(itemp,(1+j+(i-1)*ntgts))=1.0
300      continue
320   continue
      return
      end
c
c
c
      subroutine filein
      common/main/ ntof(10),iprin(61),isub(10,20),itype(10,20),
     1 wght(10,20),wpnyld(20),tgtval(20),ntgtpr(20),rhs(61),nprit,
     1 nhedg,numtgt(20),wpncep(20)
      common/aijin1/ nwpns,ntgts,sparse(20,20),aij(61,401),icount
      common/wtint/ wpname(20),tgtnam(20)
      character wpname(20)*6,tgtnam(20)*6
      dimension nc(10), iconnp(20,10)
c
c
c     nprit= number of priorities.
c     ntof(np)= number of terms in the objective function at priority np.
c
c
c     When everything is built and/or calculated, this routine ships all
c     of the data to the appropriate files:  Sparse, Aij, and Pagpin.
c
c
      open(11,file='SPARSE')
      open(13,file='AIJ')
      open(15,file='PAGPIN')
      rewind 11
      rewind 13
      rewind 15
c
c     Writes to Sparse.
c
```

```
      do 110 j=1,ntgts
        do 100 i=1,nwpns
  100      write(11,*) sparse(j,i)
  110 continue
c
c     Writes to Aij.
c
      ncon=ntgts+nwpns+nhedg+1
      nvar=ntgts*nwpns+1
      do 140 i=1,ncon
         do 130 j=1,nvar
  130       write(13,*) aij(i,j)
  140 continue
c
c     Writes to Pagpin
c
      write(15,200) ntgts,nwpns,icount
  200 format(3I5)
c
c     Assumed nprit is known, also, there are never any real constraints
c     in this model.
c
      nvar=nvar-1
      nrcon=0
      write(15,200) nprit,nvar,nrcon
c
c     Initialize nc(n) vector.  (Number of constraints at priority n)
c
      do 205 n=1,nprit
  205    nc(n)=0
c
c     Count the number of constraints at priority n and find the subscript
c     of the jth constraint assigned priority n [ iconnp(j,n) ].
c
      do 210 i=1,ncon
             n=iprin(i)
             nc(n)=nc(n)+1
             j=nc(n)
```

```
                iconnp(j,n)=i
  210     continue
          write(15,220) (nc(np),np=1,nprit)
  220     format(10I5)
c
c         Write the subscripts of the jth constraint at priority n.
c         There can be at most 20 constraints added per priority.
c
          do 230 n=1,nprit
                  if(nc(n).eq.0) go to 230
                  nctmp=nc(n)
                  write(15,232) (iconnp(j,n),j=1,nctmp)
  230     continue
  232     format(20I5)
c
c         Given ntof(np) from 1 to nprit, write to Pagpin.
c
          write(15,220) (ntof(np),np=1,nprit)
c
c         The following is the input of equations and objectives terms.
c         They must be entered in order of priorities 1 through 10.
c         First, the goal constraints, then the objective function.
c
          mtemp=nvar+1
          do 243 i=1,nprit
                  if(nc(i).eq.0) go to 235
                  do 234 j=1,ncon
                    if(iprin(j).eq.i) then
c
c         Writes constraints at priority i
c
                      do 233 k=1,mtemp
                           write(15,*) aij(j,k)
  233                 continue
                      endif
  234                 continue
c         Writes the objective function information for priority nprit into
c         Pagpin.
```

```
c
  235           nt1=ntof(i)
                do 240 nt=1,nt1
                  write(15,310) isub(i,nt),itype(i,nt),wght(i,nt)
  240           continue
  245   continue
c
c       Writes target names and number, and weapon names to Pagpin.
c
        do 250 j=i,ntgts
           write(15,320) tgtnam(j),numtgt(j),tgtval(j)
  250   continue
        do 260 i=1,nwpns
           write(15,330) wpname(i)
  260   continue
        close(11)
        close(13)
        close(15)
  310   format(2I5,f10.6)
  320   format(a6,i5,f7.2)
  330   format(a6)
        return
        end
c
c
c
c
        subroutine change
        dimension nc(10),ncon(20,10),ntof(10),aij(61,401),
     c  isub(10,20),itype(10,20),wght(10,20),numtgt(20),
     c  ssps(20,20),tgtdia(20),tgtde(20),wpnrel(20),
     c  wpncep(20),wpnyld(20),tempde(20),tgtval(20)
        character yes,tgtnam(20)*6,wpname(20)*6,tgtpsi(20)*4
        character vntK*4,a,b,c,d,t,tgtcat(20)
        open(11,file='PAGPIN')
        open(12,file='SPARSE')
        open(14,file='cdat')
        rewind 11
```

```
      rewind 12
      rewind 14
c
c     Read Pagpin
c
      read(11,200) ntgts,nwpns,icount
      read(11,200) nprit,nvar,nrcon
      read(11,220) (nc(np),np=1,nprit)
      nconst=0
      do 40 i=1,nprit
40          nconst=nconst + nc(i)
      do 50 n=1,nprit
         if(nc(n).eq.0) go to 50
         nctmp=nc(n)
         read(11,232)(ncon(j,n),j=1,nctmp)
50    continue
      read(11,220) (ntof(np),np=1,nprit)
196   format(a4)
197   format(4a1)
198   format(i2,a1,i1)
199   format(i4)
200   format(3i5)
220   format(10i5)
232   format(20i5)
      mtemp=nvar+1
      do 245 i=1,nprit
         if (nc(i).eq.0) go to 235
         do 234 j=1,nc(i)
              itemp=ncon(j,i)
              do 233 mn=1,mtemp
                 read(11,*) aij(itemp,mn)
233           continue
234      continue
235      nt1=ntof(i)
         do 240 nt=1,nt1
              read(11,310) isub(i,nt),itype(i,nt),wght(i,nt)
240      continue
245   continue
```

```
      do 250 j=1,ntgts
         read(11,320) tgtnam(j),numtgt(j),tgtval(j)
250   continue
      do 260 i=1,nwpns
         read(11,330) wpname(i)
260   continue
310   format(2i5,f10.6)
320   format(a6,i5,f7.2)
330   format(a6)
c
c     Read SPARSE
c
      do 410 j=1,ntgts
         do 400 i=1,nwpns
              read(12,*) ssps(j,i)
400      continue
410   continue
c
c     Read cdat
c
      do 490 j=1,ntgts
         read(14,480) tgtpsi(j),tgtdia(j),tgtde(j),tgtcat(j)
490   continue
480   format(a4,f7.3,f4.2,a1)
      do 495 i=1,nwpns
         read(14,*) wpnrel(i),wpncep(i),wpnyld(i)
495   continue
      close (11)
      close (12)
      close (14)
      write(6,500)
      write(6,*)'This section permits minor changes to your original'
      write(6,*)'problem.  After the changes have been made, the '
      write(6,*)'problem should be rerun.  The following is a menu'
      write(6,*)'of options available.  (Hit return to continue)'
      read(5,4) yes
4     format(a1)
5     write(6,500)
```

```
500     format(/////////////////////)
        write(6,*)'                    MENU'
        write(6,*)
        write(6,*)'              SENSITIVITY RERUN'
        write(6,*)
        write(6,*)'     1) Change DE goals'
        write(6,*)'     2) Change weapon availability'
        write(6,*)'     3) Change target weights'
        write(6,*)'     4) Change target parameters'
        write(6,*)'     5) Change weapon parameters'
        write(6,*)
        write(6,*)'Enter your choice.  Type 0 if you are finished.'
10      read(5,*) nmenu
        if (nmenu.eq.0) then
          write(6,*)
          write(6,*)'Do you want to suppress the expanded output in
     c file alloc? (y/n)'
15          read(5,4) yes
            if(yes.ne.'y' .and. yes.ne.'n') go to 15
            if(yes.eq.'y') nrcon=1
            go to 1600
        endif
        if (nmenu.lt.0 .or. nmenu.gt.5) go to 10
        go to (1100,1200,1300,1400,1500) nmenu
c
c       DE goals
c
1100    do 1105 j=1,ntgts
           tempde(j)=tgtde(j)
1105    continue
        call dechek(tgtde,tgtnam,ntgts,tgtcat)
        do 1110 i=1,ntgts
           aij(i,1)=-1*numtgt(i)*log(1-tgtde(i))
1110    continue
        if(nc(3).ne.0 .and. ntof(3).ne.ntgts) then
           do 1120 npr=3,(nprit-1)
                do 1115 ntm=1,(ntof(npr))
                   itemp=isub(npr,ntm)
```

```
            wght(npr,ntm)=wght(npr,ntm)*(1-tgtde(itemp))/(1-tempde(itemp))
1115              continue
1120           continue
            endif
            go to 5
c
c           Weapon availability
c
1200        write(6,500)
            write(6,*)' NO.   WPN NAME   NO. AVAILABLE'
            write(6,*)
            do 1210 i=1,nwpns
               itemp=ntgts+i
               write(6,1220) i,wpname(i),aij(itemp,1)
1210        continue
1220        format(2x,i2,7x,a6,6x,f6.1)
            write(6,*)
            write(6,*)'Is this data O.K.? (y/n)'
1225        read(5,4) yes
            if(yes.ne.'y' .and. yes.ne.'n') go to 1225
            if(yes.eq.'y') go to 5
            write(6,*)
            write(6,*)'Enter the number of the class that needs its'
            write(6,*)'availability changed.  If finished type 0.'
1230        read(5,*)j
            if(j.lt.0 .or. j.gt.nwpns) go to 1230
            if(j.eq.0) go to 1200
            itemp=ntgts+j
            write(6,*)'Enter the new availability.'
1235        read(5,*) aij(itemp,1)
            if(aij(itemp,1).lt.0.0) go to 1235
            write(6,*)
            write(6,*)'Enter another class number.  Type 0 if finished.'
            go to 1230
c
c           Target weight changes
c
1300        write(6,500)
```

```
      if(nc(3).eq.0 .and. ntof(3).eq.ntgts) then
        write(6,*)'Your objective function bounded the problem with'
        write(6,*)'DE goals.  This allowed all of your goals to be '
        write(6,*)'met.  Therefore, any change in target weights will'
        write(6,*)'not affect your solution.  (Hit return to get back'
        write(6,*)'to menu)'
        read(5,4) yes
      else
        write(6,*)'This section was developed to research a solution'
        write(6,*)'to the objective function problem.  Weights, not'
        write(6,*)'target values are reflected here.  See the thesis'
        write(6,*)'for more explanation.  (Hit return to continue)'
        read(5,4) yes
      endif
1301  write(6,500)
      write(6,*)' NO.   TGT NAME     WEIGHT'
      write(6,*)
      do 1310 j=1,ntgts
         do 1305 npr=3,(nprit-1)
              do 1303 ntm=1,(ntof(npr))
                if(isub(npr,ntm).eq.j) then
                 write(6,1315)j,tgtnam(j),wght(npr,ntm)
                 go to 1310
                endif
1303            continue
1305       continue
1310  continue
1315  format(2x,i2,6x,a6,4x,f7.4)
      write(6,*)
      write(6,*)'Is this data O.K.? (y/n)'
1325  read(5,4) yes
      if(yes.ne.'y' .and. yes.ne.'n') go to 1325
      if(yes.eq.'y') go to 5
      write(6,*)'Enter the number of the class that needs its weight'
      write(6,*)'changed.  If finished type 0.'
1330  read(5,*) j
      if(j.lt.0 .or. j.gt.nwpns) go to 1330
      if(j.eq.0) go to 1301
```

```
      write(6,*)'Enter the new weight for target class ',j,'.'
1335  read(5,*) temp
      if(temp.lt.0.0) temp=0.0
      do 1345 npr=3,(nprit-1)
         do 1340 ntm=1,(ntof(npr))
            if(isub(npr,ntm).eq.j) then
              wght(npr,ntm)=temp
              go to 1350
            endif
1340     continue
1345  continue
1350  write(6,*)'Enter another class number.  Type 0 if finished.'
      go to 1330
c
c     TARGET PARAMETERS
c
1400  write(6,500)
      write(6,*)' NO.    NAME      VUL    DIAMETER'
      write(6,*)
      do 1410 j=1,ntgts
         write(6,1420) j,tgtnam(j),tgtpsi(j),tgtdia(j)
1410  continue
1420  format(2x,i2,6x,a6,2x,a4,5x,f5.2)
      write(6,*)
      write(6,*)'Is this data O.K.? (y/n)'
1425  read(5,4) yes
      if(yes.ne.'y' .and. yes.ne.'n') go to 1425
      if(yes.eq.'y') go to 5
      write(6,*)'Enter the number of the class that needs its VUL or'
      write(6,*)'DIAMETER changed.  If finished type 0.'
1430  read(5,*) j
1435  format(a4)
      if(j.lt.0 .or.j.gt.ntgts) go to 1430
      if(j.eq.0) go to 1400
      write(6,*)'Enter the desired VUL (4 characters) for target class
     c',j,'.'
      read(5,1435)tgtpsi(j)
      write(6,*)'Enter the desired DIAMETER (NM) for target class ',j,
```

```
      c',
          read(5,*) tgtdia(j)
          if(tgtdia(j).lt.0.0) tgtdia(j)=0.0
          open(15,file='vntkch')
          do 1445 i=1,nwpns
            if (ssps(j,i) .gt. .999) go to 1445
            temp=-1*log(ssps(j,i))
            rewind 15
            vntk=tgtpsi(j)
            write(15,196) vntk
            rewind 15
            read(15,197) a,b,c,d
            rewind 15
            if(c.eq.'q' .or. c.eq.'0' .or. c.eq.'p' .or.
     1         c.eq.'P') then
                  read(15,198) in,t,ik
                  vn=real(in)
                  r95=tgtdia(j)/2
                  tpk=vtk(vn,t,ik,f95,wpnyld(i),wpncep(i))
            else
                  read(15,199) npsi
                  psi=real(npsi)
                  tpk=pk(wpnyld(i),psi,wpncep(i),tgtdia(j))
            endif
            ssps(j,i)=1-wpnrel(i)*tpk
            if (ssps(j,i).lt.,001) ssps(j,i)=.001
            aij(j,(1+j+(i-1)*ntgts))=-1*alog(ssps(j,i))
      c
      c       Check to see if coefficient was used in a hedging constraint.
      c       If so, it also must be changed.
      c
            do 1440 isub1 = (ntgts+nwpns+2),nconst
                  taij=aij(isub1,(1+j+(i-1)*ntgts))
                  if(taij.lt.(temp+.0001) .and. taij.gt.(temp-.0001)) then
                   aij(isub1,(1+j+(i-1)*ntgts))=-1*alog(ssps(j,i))
                  endif
      1440      continue
      1445    continue
```

```
      close (15)
      write(6,*)
      write(6,*)'Enter another class number.  Type 0 if finished.'
      go to 1430
c
c     Weapon parameter change
c
1500  write(6,500)
      write(6,*)' NO.    WPN NAME    RELIABILITY    CEP  YIELD'
      write(6,*)
      do 1510 i=1,nwpns
        write(6,1520)i,wpname(i),wpnrel(i),wpncep(i),wpnyld(i)
1510  continue
1520  format(2x,i2,8x,a6,4x,f5.2,7x,f5.2,2x,f5.2)
      write(6,*)
      write(6,*)'Is this data O.K.? (y/n)'
1525  read(5,4) yes
      if(yes.ne.'y' .and. yes.ne.'n') go to 1525
      if(yes.eq.'y') go to 5
      write(6,*)'Enter the number of the class that needs its'
      write(6,*)'reliability, CEP, or yield changed.  Type 0 '
      write(6,*)'if finished.'
1530  read(5,*)i
      if(i.lt.0 .or. i.gt.nwpns) go to 1530
      if(i.eq.0) go to 1500
      write(6,*)'Enter the desired reliability, cep, and yield for'
      write(6,*)'weapon class ',i,'.'
1535  read(5,*) wpnrel(i),wpncep(i),wpnyld(i)
      if(wpnrel(i).lt.0.0) wpnrel(i)=0.0
      if(wpncep(i).lt.0.0) wpncep(i)=0.0
      if(wpnyld(i).lt.0.0) wpnyld(i)=0.0
      open(15,file='vntkch')
      do 1545 j=1,ntgts
        if (ssps(j,i) .gt. .999) go to 1545
        temp=-1*alog(ssps(j,i))
        rewind 15
        vntk=tgtpsi(j)
        write(15,196) vntk
```

```
                rewind 15
                read(15,197) a,b,c,d
                rewind 15
                if(c.eq.'q' .or. c.eq.'0' .or. c.eq.'p' .or.
     1             c.eq.'P') then
                   read(15,198) in,t,ik
                   vn=real(in)
                   r95=tgtdia(j)/2
                   tpk=vtk(vn,t,ik,r95,wpnyld(i),wpncep(i))
                else
                      read(15,199) npsi
                      psi=real(npsi)
                      tpk=pk(wpnyld(i),psi,wpncep(i),tgtdia(j))
                endif
                ssps(j,i)=1-wpnrel(j)*tpk
                if (ssps(j,i).lt..001) ssps(j,i)=.001
                aij(j,(1+j+(i-1)*ntgts))=-1*alog(ssps(j,i))
      c
      c         Check to see if the coefficient was used in a hedging constraint
220   c
                do 1540 isub1= (ntgts+nwpns+2),nconst
                  taij=aij(isub1,(1+j+(i-1)*ntgts))
                  if(taij.lt.(temp+.0001) .and. taij.gt.(temp-.0001)) then
                   aij(isub1,(1+j+(i-1)*ntgts))=-1*alog(ssps(j,i))
                  endif
      1540      continue
      1545    continue
              close (15)
              write(6,*)
              write(6,*)'Enter another class number.  Type 0 if finished.'
              go to 1530
      c
      c       Ship all new information to AIJ2, SPAR2, and PAGP2
      c
      1600    write(6,500)
              write(6,*)'                                        PLEASE STANDBY'
              write(6,1605)
      1605    format(////////////)
```

```
      open(11,file='SPAR2')
      open(12,file='AIJ2')
      open(15,file='PAGP2')
      rewind 11
      rewind 12
      rewind 15
      read(15,200)i,j,icount
      rewind 15
      do 1620 j=1,ntgts
        do 1610 i=1,nwpns
           write(11,*) ssps(j,i)
1610    continue
1620  continue
c
c     Count the number of constraints.
c
      numcon=0
      do 1630 m=1,nprit
        numcon=numcon+nc(m)
1630  continue
      numvar=ntgts*nwpns+1
      do 1640 i=1,numcon
        do 1635 j=1,numvar
          write(12,*) aij(i,j)
1635    continue
1640  continue
c
c     Write to PAGP2
c
      icount=icount+1
      write(15,200) ntgts,nwpns,icount
      write(15,200) nprit,nvar,nrcon
      write(15,220) (nc(np),np=1,nprit)
      do 1650 n=1,nprit
        if(nc(n).eq.0) go to 1650
        nctmp=nc(n)
        write(15,232) (ncon(j,n),j=1,nctmp)
1650  continue
```

```
      write(15,220) (ntof(np),np=1,nprit)
      mtemp=nvar+1
      do 1660 i=1,nprit
         if(nc(i).eq.0) go to 1655
         do 1654 j=1,nc(i)
           itemp=ncon(j,i)
           do 1653 mn=1,mtemp
              write(15,*) aij(itemp,mn)
1653       continue
1654     continue
1655     nt1=ntof(i)
         do 1656 nt=1,nt1
              write(15,310) isub(i,nt),itype(i,nt),wght(i,nt)
1656     continue
1660  continue
      do 1670 j=1,ntgts
        write(15,320) tgtnam(j),numtgt(j),tgtval(j)
1670  continue
      do 1680 i=1,nwpns
222     write(15,330) wpname(i)
1680  continue
      write(6,1605)
      close (11)
      close (12)
      close (15)
      return
      end
c
c
c
      subroutine header
      write(6,*)'WELCOME TO'
      write(6,*)
      write(6,*)
      write(6,*)'          B B B B B       R R R R R       II
     1     K       K'
      write(6,*)'          B       B       R       R       II
     1     K     K'
```

```
      write(6,*)'          B        B      R        R      II
     1     K   K'
      write(6,*)'          B        B      R        R      II
     1     K K'
      write(6,*)'          B B B B B      R R R R R      II
     1     K'
      write(6,*)'          B        B      R        R      II
     1     K K'
      write(6,*)'          B        B      R        R      II
     1     K   K'
      write(6,*)'          B        B      R        R      II
     1     K     K'
      write(6,*)',         B B B B B      R        R      II
     1     K       K'
      write(6,*)
      write(6,*)
      write(6,*)'                              ' AN'
      write(6,*)
      write(6,*)'                            INTERACTIVE'
      write(6,*)
      write(6,*)'                         GOAL PROGRAMMING MODEL'
      write(6,*)
      write(6,*)'                              FOR'
      write(6,*)
      write(6,*)'                         NUCLEAR EXCHANGE PROBLEMS'
      do 50 i=1,450000
      i=i+1
50    continue
      write(6,1010)
1010  format(////////////////////)
      return
      end


c
c
c
      function vtk(vn,type,K1,r95,y,cep)
```

```
      integer h
      character type
      dimension a(7,2,2),b(4,2),d(4,10,2),pK(2)
      data a /
     p  8.214, -.1118,5.265e-4,2.162e-5,-6.638e-7,7.132e-9,-3.064e-11,
     a  8.783, -.1355,2.355e-3,-2.086e-4,9.901e-6,-1.872e-7,1.227e-09,
     q  8.315, -.1033,-7.908e-4,-9.039e-5,1.458e-5,-5.220e-7,5.726e-9,
     a  8.789, -.1120,-6.658e-5,-5.803e-4,5.853e-5,-1.905e-6,2.056e-8/
      data b /
     p  1.66904, -2.17442, .260926, -.0752178,
     q  1.65946, -2.15466, .295853, -.0484718/
      data d / 4*0.,
     1  -.577874, 1.56916, .0013762, .0063101,
     2  -1.22391, 3.32978, -.0020688,-.0469539,
     3  -1.957,   5.35163, -.0477323,-.154546,
     4  -2.80456, 7.74241, -.225298, -.345665,
     5  -3.81168, 10.7222, -.841780, -.513504,
     6  -5.05081, 14.6336, -2.37475, -.476885,
     7  -6.65796, 20.1955, -5.98136,  .315062,
     8  -8.92601, 28.9801, -14.2174,  3.11471,
     9  -12.7265, 45.9954, -35.5644,  12.2164, 4*0.,
     1  -.288917, .798865, -.0399065, .0011525,
     2  -.611921, 1.72876, -.186910, .0095419,
     3  -.978187, 2.83679, -.511558, .0514480,
     4  -1.40112, 4.19642, -1.13330, .181180,
     5  -1.90063, 5.91548, -2.23005, .484749,
     6  -2.50907, 8.17913, -4.11606, 1.12029,
     7  -3.28374, 11.3172, -7.35613, 2.37622,
     8  -4.34296, 16.0486,  -13.2112, 4.90666,
     9  -6.01000, 24.4300, -25.3672, 10.643/
      r=6080*r95
      c=sqrt((cep*6080)**2 +.231*r**2)
      if (c.lt..001) then
              vtK=1.
              return
      endif
      K=K1+1
      t=1
```

```
            if(type.eq.'0' .or. type.eq.'q') t=2
            xd=(y*1000)**(-1./3.)
            v=vn+d(1,K,t)+xd*(d(2,K,t)+xd*(d(3,K,t)+xd*d(4,K,t)))
            do 150 h=1,2
            s=exp(a(1,h,t)+v*(a(2,h,t)+v*(a(3,h,t)+v*(a(4,h,t)+v*(a(5,h,t)+
           &      v*(a(6,h,t)+v*a(7,h,t))))))
            u=s/(xd*c)
            if (u.gt.10.) then
                    pk(h)=1.
                    go to 150
            endif
            if (u.lt.0.1) then
                    pk(h)=0.
                    go to 150
            endif
            q=b(1,t)+u*(b(2,t)+u*(b(3,t)+u*b(4,t)))
            pk(h)=exp(-exp(q))
150         continue
            if (pk(1).gt.pk(2)) then
            vtk=pk(1)
            else
            vtk=pk(2)
            endif
            return
            end
c
c
c
c
      SUBROUTINE BOUT
      COMMON TT(10,522),TB(61),TE(61,522),TL(61,10),TA(10),TI(10,522),JC
     1OL(522,2),NCOLI,NROWI,NPRIC,NC(10),JROW(61,2),NVAR,NPRIT,IND(522)
      COMMON /PHASE1/ W,NRCON,NDVR
      COMMON /CHNG/ NCON(61,10),NTOF(10)
      INTEGER ALTST
C
C **** NPRIT=THE TOTAL NUMBER OF PRIORITIES
C ****
```

```
C **** NVAR=THE TOTAL NUMBER OF DECISION VARIABLES(INCLUDING SLACK AND
C ****       SURPLUS VARIABLES BUT EXCLUDING ARTIFICIAL VARIABLES FOR THE
C ****       REAL CONSTRAINTS)
C ****
C **** NRCON=THE NUMBER OF REAL CONSTRAINTS
C
      NRCON=0
      OPEN(11,FILE='PAGP2')
      REWIND 11
      READ(11,120) NTGTS,NWPNS,ICOUNT
      IF(ICOUNT.EQ.1) THEN
      CLOSE (11)
      OPEN (11,FILE='PAGPIN')
      OPEN (12,FILE='SPARSE')
      OPEN (13,FILE='AIJ')
      REWIND 11
      REWIND 12
      REWIND 13
      WRITE(6,*)'I AM THINKING.  PLEASE BE PATIENT.'
      READ (11,120) NTGTS,NWPNS,ICOUNT
      ELSE
      OPEN(12,FILE='SPAR2')
      OPEN(13,FILE='AIJ2')
      REWIND 12
      REWIND 13
      WRITE(6,*)'I AM THINKING.  PLEASE BE PATIENT.'
      ENDIF
      READ (11,120) NPRIT,NVAR,NOUTP
      READ (11,121) (NC(NP),NP=1,NPRIT)
      DO 101 NP=1,NPRIT
         IF (NC(NP).EQ.0) GO TO 101
         NCTMP=NC(NP)
         READ (11,122) (NCON(N,NP),N=1,NCTMP)
  101 CONTINUE
      READ (11,121) (NTOF(NP),NP=1,NPRIT)
C
C **** INITIALIZE SUBPROBLEM DIMENSIONS AND COLUMN INDICATORS.
C ****
```

```
C **** NCOLI=THE NUMBER OF COLUMNS IN THE CURRENT WORKING TABLEAU
C ****
C **** NROWI=THE NUMBER OF ROWS IN THE CURRENT WORKING TABLEAU
C ****
C **** NPRIC=THE PRIORITY CURRENTLY BEING OPTIMIZED
C ****
C **** ZERO THE TE, TL, TT, AND TI ARRAYS.
C
      NCOLI=0
      NROWI=0
      NPRIC=0
      DO 104 NCR=1,522
         IND(NCR)=1
         DO 102 NR=1,61
  102    TE(NR,NCR)=0.
         DO 103 NP=1,10
            TI(NP,NCR)=0.
  103    TT(NP,NCR)=0.
  104 CONTINUE
      DO 105 NR=1,61
      DO 105 NP=1,10
  105 TL(NR,NP)=0.
C
C **** CHECK FOR REAL CONSTRAINTS.
C
      IF (NRCON.EQ.0) GO TO 106
      CALL PHSE1
      IF (NDVR.LE.0) GO TO 116
      IF (W.GT.0.) GO TO 117
C
C **** THE PARTITIONING ALGORITHM BEGINS.
C
  106 NPRIC=NPRIC+1
      IF (NPRIC.EQ.1.AND.NRCON.EQ.0) GO TO 107
      GO TO 108
  107 CALL READ1
      GO TO 109
  108 CALL READ2
```

```
  109 CALL CINDX
      CALL TEST (NEVC,NDVR)
C
C **** IF NEVC IS LESS THAN ZERO, THE SUBPROBLEM IS OPTIMIZED.
C
      IF (NEVC.LE.0) GO TO 110
C
C **** IF NDVR IS LESS THAN ZERO, NO MINIMUM POSITIVE RATIO WAS FOUND.
C
      IF (NDVR.LE.0) GO TO 116
      CALL PERM (NEVC,NDVR)
      GO TO 109
C
C **** IF THERE ARE NO MORE PRIORITIES, TOTAL PROBLEM IS OPTIMIZED.
C **** PRINT THE OPTIMAL SOLUTION.
C
  110 IF (NPRIC.EQ.NPRIT) GO TO 115
C
C **** SINCE THERE ARE MORE PRIORITIES, MOVE ON TO THE NEXT SUBPROBLEM
C **** IF THERE ARE ALTERNATE SOLUTIONS.  FIRST, ELIMINATE THOSE
C **** COLUMNS WHICH CAN NOT ENTER THE BASIS.  IF THERE ARE NO
C **** ALTERNATE SOLUTIONS, PRINT THE UNIQUE OPTIMAL SOLUTION.
C
      ALTST=0
      DO 112 NCR=1,NCOLI
         IF (IND(NCR).EQ.0) GO TO 112
         IF (TI(NPRIC,NCR).GT.0.) GO TO 112
         DO 111 NR=1,NROWI
            IF (JROW(NR,1).EQ.JCOL(NCR,1).AND.JROW(NR,2).EQ.JCOL(NCR,2))
     1       GO TO 112
  111    CONTINUE
         ALTST=1
  112 CONTINUE
C
C **** IF ALTST=1, THERE ARE ALTERNATE SOLUTIONS.
C
      IF (ALTST.EQ.1) GO TO 113
      GO TO 115
```

```
C
C **** ELIMINATE THOSE COLUMNS WITH A POSITIVE RELATIVE COST AT
C **** PRIORITY NPRIC.
C
  113 DO 114 NCR=1,NCOLI
  114 IF (TI(NPRIC,NCR).GT.0.) IND(NCR)=0
      GO TO 106
C
C **** THE OPTIMIZATION IS OVER. PRINT OUT THE FINAL SOLUTION.
C
  115 CALL POUT(NTGTS,NWPNS,ICOUNT,NOUTP)
      GO TO 119
  116 WRITE (6,123) NPRIC
      GO TO 119
  117 WRITE (6,124) W
      WRITE (6,125)
      DO 118 NR=1,NROWI
         WRITE (6,126) JROW(NR,1),JROW(NR,2),TB(NR)
  118 CONTINUE
  119 RETURN
C
  120 FORMAT (3I5)
  121 FORMAT (10I5)
  122 FORMAT (20I5)
  123 FORMAT (/  40H   THE PROGRAM TERMINATED ON SUBPROBLEM ,I4, 42H  NO
     1 MINIMUM POSITIVE RATIO COULD BE FOUND)
  124 FORMAT (/  65H  THE PROGRAM TERMINATED IN PHASE 1 WITH OBJECTIVE F
     1UNCTION VALUE,F15.4)
  125 FORMAT (/  55H  THE OPTIMAL SOLUTION TO THE PHASE 1 PROBLEM IS
     1    //   6H  TYPE,2X,  3HSUB,8X,  5HVALUE)
  126 FORMAT (2I5,F15.4)
C
      END
      SUBROUTINE PHSE1
C
C **** SUBROUTINE PHSE1 READS IN ANY REAL CONSTRAINTS AND PERFORMS A
C **** PHASE 1 SIMPLEX PROCEDURE IN ORDER TO FIND AN INITIAL BASIC
C **** FEASIBLE SOLUTION.
```

```
C
      COMMON TT(10,522),TB(61),TE(61,522),TL(61,10),TA(10),TI(10,522),JC
     10L(522,2),NCOLI,NROWI,NPRIC,NC(10),JROW(61,2),NVAR,NPRIT,IND(522)
      COMMON /PHASE1/ W,NRCON,NDVR
      DIMENSION C(512), CR(512), CB(56)
C
C **** SET COLUMN AND ROW HEADINGS
C
      DO 101 NV=1,NVAR
         JCOL(NV,1)=2
  101 JCOL(NV,2)=NV
      DO 102 NR=1,NRCON
         JROW(NR,1)=1
         JROW(NR,2)=NR
         NAR=NVAR+NR
         JCOL(NAR,1)=1
  102 JCOL(NAR,2)=NR
C
C **** READ IN COEFFICIENTS AND RHS OF REAL CONSTRAINTS
C
      DO 103 NR=1,NRCON
         READ (11,118) TB(NR),(TE(NR,NV),NV=1,NVAR)
  103 CONTINUE
C
C **** PUT IDENTITY MATRIX IN FOR ARTIFICIAL VARIABLES
C
      DO 104 NR=1,NRCON
         NAR=NVAR+NR
  104 TE(NR,NAR)=1.
C
C **** SET C(J)=0 FOR ALL DECISION VARIABLES AND C(J)=1 FOR ALL
C **** ARTIFICIAL VARIABLES
C
      DO 105 NV=1,NVAR
  105 C(NV)=0.
      DO 106 NR=1,NRCON
         CB(NR)=1.
         NAR=NVAR+NR
```

```
  106 C(NAR)=1.
C
C **** CALCULATE RELATIVE COST COEFFICIENTS CR(.)
C
      NCOL=NVAR+NRCON
  107 DO 108 NV=1,NCOL
         CR(NV)=C(NV)
      DO 108 NR=1,NRCON
  108 CR(NV)=CR(NV)-CB(NR)*TE(NR,NV)
C
C **** CHECK FOR OPTIMALITY
C
      VEVC=0.
      NEVC=0
      DO 109 NCO=1,NCOL
         NV=NCO
         IF (CR(NV).GE.0.) GO TO 109
         IF (CR(NV).GE.VEVC) GO TO 109
         VEVC=CR(NV)
         NEVC=NV
  109 CONTINUE
C
C **** IF NEVC=0, PHASE 1 IS OPTIMIZED.  CALCULATE OBJECTIVE FUNCTION.
C
      IF (NEVC.EQ.0) GO TO 115
C
C **** DETERMINE DEPARTING VARIABLES ROW.
C
      NDVR=0
      VDVR=10.0E+20
      DO 111 NRI=1,NRCON
         NR=NRI
         IF (TE(NR,NEVC).LE.0.) GO TO 111
         V=TB(NR)/TE(NR,NEVC)
         IF (NDVR.EQ.0) GO TO 110
         IF (V-VDVR) 110,110,111
  110    VDVR=V
         NDVR=NR
```

```
  111 CONTINUE
C
C **** IF NDVR=0, MINIMUM RATIO RULE FAILED.  RETURN.
C
      IF (NDVR.EQ.0) RETURN
C
C **** PERFORM THE PIVOT.  REPLACE HEADINGS AND COST COEFFICIENT.
C
      JROW(NDVR,1)=JCOL(NEVC,1)
      JROW(NDVR,2)=JCOL(NEVC,2)
      CB(NDVR)=C(NEVC)
C
C **** COMPUTE NEW TE ARRAY
C
      PIV=TE(NDVR,NEVC)
      PIB=TB(NDVR)
      DO 113 NR=1,NRCON
         IF (NR.EQ.NDVR) GO TO 113
         IF (ABS(TE(NR,NEVC)).LE.0.0005) GO TO 113
         PIX=TE(NR,NEVC)/PIV
         TB(NR)=FIX(TB(NR)-PIX*PIB)
         DO 112 NV=1,NCOL
  112    TE(NR,NV)=FIX(TE(NR,NV)-TE(NDVR,NV)*PIX)
  113 CONTINUE
      TB(NDVR)=FIX(PIB/PIV)
      DO 114 NV=1,NCOL
  114 TE(NDVR,NV)=FIX(TE(NDVR,NV)/PIV)
C
C **** END OF PIVOT OPERATIONS.  PROCEED TO NEXT ITERATION.
C
      GO TO 107
C
C **** CALCULATE W, THE PHASE-1 OBJECTIVE FUNCTION.
C
  115 W=0.
      DO 116 NR=1,NRCON
  116 W=W+TB(NR)*CB(NR)
C
```

```
C **** INITIALIZE THOSE PORTIONS OF THE TABLEAU ASSIGNED TO THE
C **** ARTIFICIAL VARIABLES.
C
      DO 117 NR=1,NRCON
      DO 117 NV=1,NCOL
         IF (NV.LE.NVAR) GO TO 117
         TE(NR,NV)=0.
  117 CONTINUE
C
C **** UPDATE NCOLI AND NROWI PARAMETERS.
C
      NROWI=NRCON
      NCOLI=NVAR
      RETURN
C
  118 FORMAT (8F10.0)
C
      END
      SUBROUTINE READ1
C
C **** SUBROUTINE READ1 READS IN THE GOAL CONSTRAINTS AND OBJECTIVE
C **** FUNCTION TERMS ASSIGNED TO PRIORITY ONE.
C **** SUBROUTINE READ1 IS NOT USED IF REAL CONSTRAINTS ARE PRESENT.
C
      COMMON TT(10,522),TB(61),TE(61,522),TL(61,10),TA(10),TI(10,522),JC
     1OL(522,2),NCOLI,NROWI,NPRIC,NC(10),JROW(61,2),NVAR,NPRIT,IND(522)
      COMMON /CHNG/ NCON(61,10),NTOF(10)
C
C **** SET COLUMN AND ROW HEADINGS.
C
      DO 101 NV=1,NVAR
         JCOL(NV,1)=2
  101 JCOL(NV,2)=NV
      NC11=NC(1)
      DO 102 NCR=1,NC11
         NC1=NVAR+2*NCR-1
         NC2=NVAR+2*NCR
         JCOL(NC1,1)=4
```

```
            JCOL(NC1,2)=NCON(NCR,1)
            JCOL(NC2,1)=3
            JCOL(NC2,2)=NCON(NCR,1)
            JROW(NCR,1)=4
  102 JROW(NCR,2)=NCON(NCR,1)
C
C **** READ IN THE GOAL CONSTRAINTS ASSIGNED TO PRIORITY 1.
C
      NC1=NC(1)
      WRITE(6,*)'I AM WORKING ON PRIORITY  1 .'
      DO 103 NCR=1,NC1
         NV1=NVAR+2*NCR-1
         NV2=NVAR+2*NCR
         READ (11,*) TB(NCR)
      DO 108 NV=1,NVAR
         READ (11,*) TE(NCR,NV)
  108 CONTINUE
C
C **** PUT +1 IN FOR D- AND -1 IN FOR D+
C
            TE(NCR,NV1)=1.
            TE(NCR,NV2)=-1.
  103 CONTINUE
      NCOLI=NV2
      NROWI=NC(1)
C
C **** READ IN THE OBJECTIVE FUNCTION TERMS FOR PRIORITY 1.
C
      NT1=NTOF(1)
      DO 104 NT=1,NT1
         READ (11,106) ISUB,ITYPE,WGHT
         CALL PLACE (ISUB,ITYPE,WGHT)
  104 CONTINUE
      RETURN
C
  106 FORMAT (2I5,F10.6)
C
      END
```

```
      SUBROUTINE READ2
C
C **** SUBROUTINE READ2 READS IN THE GOAL CONSTRAINTS AND OBJECTIVE
C **** FUNCTION TERMS ASSIGNED TO PRIORITY NPRIC.
C **** SUBROUTINE READ2 IS ALSO USED TO READ IN THE FIRST PRIORITY GOAL
C **** CONSTRAINTS AND OBJECTIVE FUNCTION TERMS IF REAL CONSTRAINTS ARE
C **** PRESENT.
C
      COMMON TT(10,522),TB(61),TE(61,522),TL(61,10),TA(10),TI(10,522),JC
     1OL(522,2),NCOLI,NROWI,NPRIC,NC(10),JROW(61,2),NVAR,NPRIT,IND(522)
      COMMON /CHNG/ NCON(61,10),NTOP(10)
      WRITE(6,*)'I AM WORKING ON PRIORITY ',NPRIC,'.'
      IF (NC(NPRIC).EQ.0) GO TO 107
C
C **** READ IN THE COEFFICIENTS OF THE X'S.
C
      NCTMP=NC(NPRIC)
      DO 106 NRI=1,NCTMP
         NR=NRI+NROWI
         NC1=NCOLI+2*NRI-1
         NC2=NCOLI+2*NRI
         JCOL(NC1,1)=4
         JCOL(NC1,2)=NCON(NRI,NPRIC)
         JCOL(NC2,1)=3
         JCOL(NC2,2)=NCON(NRI,NPRIC)
         READ (11,*) TB(NR)
         DO 111 NV=1,NVAR
            READ(11,*) TE(NR,NV)
  111    CONTINUE
         TE(NR,NC1)=1.
         TE(NR,NC2)=-1.
C
C **** PERFORM THE ROW REDUCTION.
C
         DO 102 NRC=1,NROWI
            IF (JROW(NRC,1).NE.2) GO TO 102
            J=JROW(NRC,2)
            TB(NR)=TB(NR)-TE(NR,J)*TB(NRC)
```

```
            DO 101 NCR=1,NC2
               IF (NCR.EQ.J) GO TO 101
               TE(NR,NCR)=TE(NR,NCR)-TE(NR,J)*TE(NRC,NCR)
  101       CONTINUE
            TE(NR,J)=0.
  102    CONTINUE
C
C **** DETERMINE THE DEVIATIONAL VARIABLE TO ENTER THE BASIS.
C
         IF (TB(NR)) 103,105,105
C
C **** SINCE TB IS LESS THAN ZERO, MULTIPLY THE ROW BY -1 AND ENTER D+
C **** IN THE BASIS.
C
  103    DO 104 NCR=1,NC2
  104    TE(NR,NCR)=-TE(NR,NCR)
         TB(NR)=-TB(NR)
         JROW(NR,1)=3
         JROW(NR,2)=NCON(NRI,NPRIC)
         GO TO 106
C
C **** SINCE TB IS GREATER THAN OR EQUAL TO ZERO ENTER D- IN THE BASIS.
C
  105    JROW(NR,1)=4
         JROW(NR,2)=NCON(NRI,NPRIC)
  106 CONTINUE
C
C **** INCREASE THE PARAMETERS NCOLI AND NROWI.
C
      NCOLI=NC2
      NROWI=NR
C
C **** READ IN THE OBJECTIVE FUNCTION TERMS FOR PRIORITY NPRIC.
C
  107 NTTMP=NTOF(NPRIC)
      DO 108 NT=1,NTTMP
         READ (11,110) ISUB,ITYPE,WGHT
         CALL PLACE (ISUB,ITYPE,WGHT)
```

```
  108 CONTINUE
      RETURN
C
  110 FORMAT (2I5,F10.6)
C
      END
      SUBROUTINE PLACE (ISUB,ITYPE,WGHT)
C
C **** SUBROUTINE PLACE PUTS THE OBJECTIVE FUNCTION WEIGHTS FOR THE
C **** DEVIATION VARIABLES AT THE CURRENT PRIORITY LEVEL (NPRIC) IN THE
C **** CORRECT POSITIONS IN THE AUGMENTED TABLEAU.
C ****
C **** ISUB=THE SUBSCRIPT OF THE DEVIATIONAL VARIABLE
C ****
C **** ITYPE=3, IF POSITIVE DEVIATIONAL VARIABLE (D+)
C ****       4, IF NEGATIVE DEVIATIONAL VARIABLE (D-)
C ****
C **** WGHT=THE CARDINAL WEIGHT OF THIS DEVIATIONAL VARIABLE AT THE
C ****      CURRENT PRIORITY LEVEL
C
      COMMON TT(10,522),TB(61),TE(61,522),TL(61,10),TA(10),TI(10,522),JC
     1OL(522,2),NCOLI,NROWI,NPRIC,NC(10),JROW(61,2),NVAR,NPRIT,IND(522)
      COMMON /CHNG/ NCON(61,10),NTOF(10)
C
C **** PLACE THE WEIGHT IN THE PROPER COLUMN IN THE TOP STUB.
C
      NC1=NVAR+1
      DO 101 NCR=NC1,NCOLI
         IF (JCOL(NCR,1).EQ.ITYPE.AND.JCOL(NCR,2).EQ.ISUB) GO TO 102
  101 CONTINUE
  102 TT(NPRIC,NCR)=WGHT
C
C **** PLACE THE WEIGHT IN THE PROPER ROW IN THE LEFT STUB.
C
      DO 103 NR=1,NROWI
         IF (JROW(NR,1).EQ.ITYPE.AND.JROW(NR,2).EQ.ISUB) GO TO 104
  103 CONTINUE
      GO TO 105
```

```
  104 TL(NR,NPRIC)=WGHT
  105 CONTINUE
      RETURN
C
      END
      SUBROUTINE CINDX
C
C **** SUBROUTINE CINDX COMPUTES THE RELATIVE COST COEFFICIENTS FOR EACH
C **** VARIABLE IN THE CURRENT TABLEAU(THE TI( , , ) ARRAY) AND THE
C **** OBJECTIVE FUNCTION VALUE(THE TA(.) ARRAY) AT THE CURRENT
C **** PRIORITY(NPRIC)
C
      COMMON TT(10,522),TB(61),TE(61,522),TL(61,10),TA(10),TI(10,522),JC
     10L(522,2),NCOLI,NROWI,NPRIC,NC(10),JROW(61,2),NVAR,NPRIT,IND(522)
C
C **** COMPUTE TA(NPRIC) AND TI(NPRIC,NC)   NC=1,....,NCOLI
C
      TA(NPRIC)=0.
      DO 101 NR=1,NROWI
  101 TA(NPRIC)=TA(NPRIC)+TB(NR)*TL(NR,NPRIC)
      DO 102 NCR=1,NCOLI
         TI(NPRIC,NCR)=TT(NPRIC,NCR)
      DO 102 NR=1,NROWI
  102 TI(NPRIC,NCR)=TI(NPRIC,NCR)-TE(NR,NCR)*TL(NR,NPRIC)
      RETURN
C
      END
      SUBROUTINE TEST (NEVC,NDVR)
C
C **** SUBROUTINE TEST DETERMINES THE NEXT ENTERING VARIABLE'S COLUMN
C **** (NEVC) AND THE NEXT DEPARTING VARIABLE'S ROW(NDVR).  IF NO
C **** FURTHER OPTIMIZATION IS POSSIBLE, THE VALUE NEVC=0 IS RETURNED.
C **** IF NDVR=0 IS RETURNED, NO MINIMUM POSITIVE RATIO COULD BE FOUND
C **** IN THE CURRENT PIVOT OPERATION,I.E., ALL OF THE COEFFICIENTS
C **** TE( . ,NEVC) ARE NONPOSITIVE.
C
      COMMON TT(10,522),TB(61),TE(61,522),TL(61,10),TA(10),TI(10,522),JC
     10L(522,2),NCOLI,NROWI,NPRIC,NC(10),JROW(61,2),NVAR,NPRIT,IND(522)
```

```
      NDVR=0
      NEVC=0
      VEVC=0.
      VDVR=10.0E+20
C
C **** DETERMINE ENTERING VARIABLE'S COLUMN.
C
      DO 101 NCR=1,NCOLI
         IF (TI(NPRIC,NCR).GE.0.) GO TO 101
         IF (IND(NCR).EQ.0) GO TO 101
         IF (TI(NPRIC,NCR).GE.VEVC) GO TO 101
         NEVC=NCR
         VEVC=TI(NPRIC,NCR)
  101 CONTINUE
C
C **** IF NEVC=0, SUBPROBLEM NPRIC IS OPTIMIZED. RETURN.
C
      IF (NEVC.EQ.0) RETURN
C
C **** DETERMINE DEPARTING VARIABLE'S ROW.
C
      DO 105 NR=1,NROWI
         IF (TE(NR,NEVC).LE.0.) GO TO 105
         V=TB(NR)/TE(NR,NEVC)
         IF (NDVR.EQ.0) GO TO 104
         IF (V-VDVR) 104,102,105
  102    DO 103 NP=1,NPRIC
            IF (TL(NR,NP)-TL(NDVR,NP)) 105,103,104
  103    CONTINUE
  104    VDVR=V
         NDVR=NR
  105 CONTINUE
      RETURN
C
      END
      SUBROUTINE PERM (NEVC,NDVR)
C
C **** SUBROUTINE PERM PERFORMS THE PIVOT OPERATION USING THE PIVOT
```

```
C **** ELEMENT IN COLUMN NEVC AND ROW NDVR AND COMPUTES THE NEW TABLEAU.
C
      COMMON TT(10,522),TB(61),TE(61,522),TL(61,10),TA(10),TI(10,522),JC
     10L(522,2),NCOLI,NROWI,NPRIC,NC(10),JROW(61,2),NVAR,NPRIT,IND(522)
C
C **** REPLACE HEADING FOR ROW NDVR.
C
      JROW(NDVR,1)=JCOL(NEVC,1)
      JROW(NDVR,2)=JCOL(NEVC,2)
C
C **** REPLACE TL VECTOR FOR ROW NDVR
C
      DO 101 NP=1,NPRIC
  101 TL(NDVR,NP)=TT(NP,NEVC)
C
C **** COMPUTE NEW TE ARRAY.
C
      PIV=TE(NDVR,NEVC)
      PIB=TB(NDVR)
      DO 103 NR=1,NROWI
         IF (NR.EQ.NDVR) GO TO 103
         IF (ABS(TE(NR,NEVC)).LE.0.0005) GO TO 103
         PIX=TE(NR,NEVC)/PIV
         TB(NR)=FIX(TB(NR)-PIX*PIB)
         DO 102 NCR=1,NCOLI
  102    TE(NR,NCR)=FIX(TE(NR,NCR)-TE(NDVR,NCR)*PIX)
  103 CONTINUE
      TB(NDVR)=FIX(PIB/PIV)
      DO 104 NCR=1,NCOLI
  104 TE(NDVR,NCR)=FIX(TE(NDVR,NCR)/PIV)
      RETURN
C
      END
      FUNCTION FIX(Z)
C
C **** FUNCTION FIX BRINGS FLOATING POINT VALUES THAT ARE WITHIN 1.E-5
C **** OF AN INTEGER TO THAT INTEGER.
C
```

```
      FIX=AINT(Z+SIGN(.5,Z))
      IF (ABS(FIX-Z).GT. 1.E-5) FIX=Z
      RETURN
C
      END
      SUBROUTINE POUT(NTGTS,NWPNS,ICOUNT,NOUTP)
C
C **** SUBROUTINE POUT PREPARES AND PRINTS THE SOLUTION INFORMATION.
C
      COMMON TT(10,522),TB(61),TE(61,522),TL(61,10),TA(10),TI(10,522),JC
     1OL(522,2),NCOLI,NROWI,NPRIC,NC(10),JROW(61,2),NVAR,NPRIT,IND(522)
      COMMON /CHNG/ NCON(61,10),NTOF(10)
      CHARACTER TGTNAM(20)*6,WPNAME(20)*6
      DIMENSION WOUT(522,4), RLHS(61,10),TGTRMN(20),WUSED(20)
      DIMENSION WLEFT(20),TGTVAL(20)
      DIMENSION DIFF(61),NUMTGT(20),RHS(61),SSPS(400),ACHIEV(20)
      OPEN (15,FILE='alloc')
      IF(ICOUNT.EQ.1) REWIND 15
      WRITE (15,860) ICOUNT
      WRITE (15,123) NPRIC,NROWI
      WRITE (6,*) 'I AM WORKING ON THE OUTPUT.'
C
C **** OUTPUT ARRAY IS ZEROED.
C
      DO 101 I=1,212
      DO 101 J=1,4
  101 WOUT(I,J)=0.
C
C **** OUTPUT ARRAY IS FILLED.
C
      DO 102 NP=1,NPRIC
  102 WOUT(NP,1)=FIX(TA(NP))
      DO 103 NR=1,NROWI
         I1=JROW(NR,1)
         I2=JROW(NR,2)
  103 WOUT(I2,I1)=FIX(TB(NR))
C
C **** IF ALL PRIORITIES HAVE BEEN INCLUDED, PRINT OPTIMAL SOLUTION.
```

```
C **** IF NOT, WE MUST CALCULATE VALUES FOR REMAINING TA'S AND D- AND D+
C
      IF (NPRIC.GE.NPRIT) GO TO 114
      NP1=NPRIC+1
      DO 113 NP=NP1,NPRIT
         WRITE(6,*)'I AM WORKING ON PRIORITY',NP,'.'
         TA(NP)=0.
         IF (NC(NP).EQ.0) GO TO 106
C
C **** READ IN THE GOAL CONSTRAINTS ASSIGNED TO PRIORITY NP.
C
         NCTMP=NC(NP)
         DO 105 NCI=1,NCTMP
            NR=NROWI+NCI
            READ (11,*) TB(NR)
            DO 255 NV=1,NVAR
                READ(11,*) TE(NR,NV)
 255        CONTINUE
            RLHS(NCI,NP)=0.
            DO 104 NV=1,NVAR
 104        RLHS(NCI,NP)=RLHS(NCI,NP)+TE(NR,NV)*WOUT(NV,2)
            DIFF(NCI)=TB(NR)-RLHS(NCI,NP)
 105     CONTINUE
C
C **** READ THE OBJECTIVE FUNCTION TERMS FOR PRIORITY NP.
C
 106     NTTMP=NTOF(NP)
         DO 112 NT=1,NTTMP
            READ (11,125) ISUB,ITYPE,WGHT
            IF (NC(NP).EQ.0) GO TO 111
            NCTMP=NC(NP)
            DO 110 NCI=1,NCTMP
               IF (ISUB.NE.NCON(NCI,NP)) GO TO 110
               IF (DIFF(NCI)) 107,108,109
 107           IF (ITYPE.NE.3) GO TO 110
               WOUT(ISUB,3)=-DIFF(NCI)
 108           GO TO 110
 109           IF (ITYPE.NE.4) GO TO 110
```

```
                WOUT(ISUB,4)=DIFF(NCI)
  110         CONTINUE
  111         TA(NP)=TA(NP)+WGHT*WOUT(ISUB,ITYPE)
  112       CONTINUE
            NROWI=NROWI+NC(NP)
C
C **** FILL IN THE OUTPUT VALUE FOR ATTAINMENT OF PRIORITY NP.
C
          WOUT(NP,1)=PIX(TA(NP))
  113 CONTINUE
C
C **** PRINT OPTIMAL SOLUTION
C
  114 WRITE (15,126)
      IF(NOUTP.EQ.1) GO TO 200
      WRITE (15,133)
      WRITE (15,134)
      I=MAXO(NPRIT,NVAR,NROWI)
      DO 121 K=1,I
         IF (K.GT.NPRIT) GO TO 119
         IF (K.GT.NVAR) GO TO 118
         WRITE (15,135) K,(WOUT(K,J),J=1,4)
         GO TO 121
  118    WRITE (15,136) K,WOUT(K,1),(WOUT(K,J),J=3,4)
         GO TO 121
  119    IF (K.GT.NVAR) GO TO 120
         WRITE (15,137) K,(WOUT(K,J),J=2,4)
         GO TO 121
  120    WRITE (15,138) K,(WOUT(K,J),J=3,4)
  121 CONTINUE
      WRITE (15,126)
  200   do 250 j=1,ntgts
          read(11,320) tgtnam(j),numtgt(j),tgtval(j)
  250   continue
        do 260 i=1,nwpns
          read(11,330) wpname(i)
  260   continue
  320   format(a6,i5,f7.2)
```

```
  330   format(a6)
c
c       Reads aij to get RHS(1--ntgts+nwpns)
c
        itvar=ntgts*nwpns
        irhs=ntgts+nwpns
        do 360 i=1,irhs
          read(13,*) rhs(i)
          do 350 j=1,itvar
  350       read(13,*) temp
  360   continue
c
c       Convert RHS(j) to goal on targets
c
        do 370 j=1,ntgts
                rhs(j)=1-exp(-1.0*rhs(j)/numtgt(j))
  370   continue
c
c       Compute DE achievement on each tgt class and total value destroyed.
c
        valdes=0.0
        do 500 j=1,ntgts
          achiev(j)=1.0
          do 400 i=1,nwpns
           read(12,*) ssps(i)
      achiev(j)=achiev(j)*ssps(i)**(wout((j+(i-1)*ntgts),2)/numtgt(j))
  400     continue
          tgtrmn(j)=achiev(j)*numtgt(j)
          achiev(j)=1-achiev(j)
          valdes=valdes+achiev(j)*tgtval(j)*numtgt(j)
  500   continue
c
c       Compute wpns used from each class
c
        do 550 i=1,nwpns
          wused(i)=0
          do 540 j=1,ntgts
  540           wused(i)=wused(i)+wout((j+(i-1)*ntgts),2)
```

```
        wleft(i)=rhs(i+ntgts)-wused(i)
550   continue
      write(6,*)
      write(6,*)
      write(15,*)
      write(6,700) valdes
      write(15,700) valdes
      write(6,*)
      write(15,*)
      write(6,800)
      write(6,810)
      write(15,800)
      write(15,810)
      ipcnt=4
      do 600 j=1,ntgts
         itemp=0
         do 590 i=1,nwpns
              if(wout((j+(i-1)*ntgts),2).gt.0.5) then
     write(6,820) tgtnam(j),wpname(i),wout((j+(i-1)*ntgts),2),rhs(j),
    1achiev(j),tgtrmn(j)
     write(15,820)tgtnam(j),wpname(i),wout((j+(i-1)*ntgts),2),rhs(j),
    1achiev(j),tgtrmn(j)
              itemp=i+1
              ipcnt=ipcnt+1
              go to 595
              endif
590      continue
     write(6,825) tgtnam(j),rhs(j),achiev(j),tgtrmn(j)
     write(15,825)tgtnam(j),rhs(j),achiev(j),tgtrmn(j)
         ipcnt=ipcnt+1
         go to 600
595      do 596 i=itemp,nwpns
              if(wout((j+(i-1)*ntgts),2).gt.0.5) then
                 write(6,830) wpname(i),wout((j+(i-1)*ntgts),2)
                 write(15,830)wpname(i),wout((j+(i-1)*ntgts),2)
                 ipcnt=ipcnt+1
              endif
596      continue
```

```
            if(ipcnt.ge.19) call page(ipcnt)
  600     continue
          call page(ipcnt)
          write(6,*)
          write(15,*)
          write(6,840)
          write(15,840)
          do 650 i=1,nwpns
            write(6,850) wpname(i),rhs(ntgts+i),wused(i),wleft(i)
            write(15,850)wpname(i),rhs(ntgts+i),wused(i),wleft(i)
          ipcnt=ipcnt+1
          if(ipcnt.ge.19)call page(ipcnt)
  650     continue
          call page(ipcnt)
          close(11)
          close(12)
          close(13)
          close(15)
        RETURN
C
  123 FORMAT (/  39H  THE OPTIMIZATION ENDED ON SUBPROBLEM ,I5 /  13H  T
     1HERE WERE ,I5, 42H CONSTRAINTS IN THE FINAL OPTIMAL TABLEAU.)
  125 FORMAT (2I5,F10.6)
  126 FORMAT (//80(1H*))
  133 FORMAT (1H0, 15H OUTPUT SUMMARY)
  134 FORMAT (1H0,  9HSUBSCRIPT,11X,  8H   A OPT,7X,  8H   X OPT,7X,  9H
     1  POS DEV,6X,  9H  NEG DEV /)
  135 FORMAT (I8,7X,4F15.4)
  136 FORMAT (I8,7X,F15.4,15X,2F15.4)
  137 FORMAT (I8,22X,3F15.4)
  138 FORMAT (I8,37X,2F15.4)
  700 FORMAT ('TOTAL VALUE DESTROYED WAS ',F7.1,'.')
  800 FORMAT (24X,'NUMBER',29X,'NUMBER')
  810 FORMAT (3X,'TGTNAM',5X,'WPNCLASS',2X,'ASSIGNED',4X,'GOAL',4X,'ACHI
     1EVEMENT',3X,'REMAINING')
  820 FORMAT (3X,A6,6X,A6,2X,F7.2,7X,F3.2,9X,F3.2,9X,F6.1)
  825 FORMAT (3X,A6,28X,F3.2,9X,F3.2,9X,F6.1)
  830 FORMAT (15X,A6,2X,F7.2)
```

```
  840 FORMAT (3X,'WPNAME',5X,'NUMBER',4X,'USED',6X,'REMAIN')
  850 FORMAT (3X,A6,4X,F6.1,4X,F6.1,4X,F6.1)
  860 FORMAT (/  10HRUN NUMBER,I2, 1H.)
C
      END
C
C
C
        subroutine page(ipcnt)
        character abc*1
        lines=20-ipcnt
        ipcnt=0
        if(lines.lt.0) lines=0
        do 10 i=1,lines
          write(6,30)' '
   10   continue
   30   format(a1)
        write(6,*) 'Hit return to continue.'
        read(5,40) abc
   40   format(a1)
        return
        end
```

VITAE

Robert E. Bunnell was born on 5 August 1953 in Kankakee, Illinois. He graduated from high school in Petersburg, Virginia, in 1971. After receiving a B.S. in Mathematics Education from Virginia Polytechnic Institute in 1975, he was commissioned into the USAF through the ROTC program. He completed Navigator Training in March 1977 and was then assigned to the KC-135 at Rickenbacker AFB, Ohio, and at Loring AFB, Maine. He graduated from the Air Force Institute of Technology in March of 1984. His permanent address is as follows:

368 Hawser Lane
Naples, Florida 33940

Richard A. Takacs was born on 20 November 1953 in San Francisco, California. He graduated from high school in Stockton, California, in 1971. He attended the US Air Force Academy, from which he graduated in 1975, receiving both a commission and a B.S. in Basic Science. He graduated from Undergraduate Pilot Training in June 1976 and was assigned to Nellis AFB, Nevada, as an Aircraft Commander in the F4D. In October 1979, he was assigned to Clark AB, Philippines, flying the F4G Wild Weasel. He graduated from AFIT in March 1984. His permanent mailing address is the following:

734 MacDuff Ave.
Stockton, California 95210

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S): AFIT/GST/OS/84M-5	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: School of Engineering; 6b. OFFICE SYMBOL (If applicable): AFIT/ENS	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code): Air Force Institute of Technology (AU) Wright-Patterson AFB, Ohio 45433	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NOS. — PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE (Include Security Classification): See Box 19	
12. PERSONAL AUTHOR(S): Robert E. Bunnell, Capt, USAF; Richard A. Takacs, Capt, USAF	
13a. TYPE OF REPORT: MS Thesis; 13b. TIME COVERED: FROM _____ TO _____; 14. DATE OF REPORT (Yr., Mo., Day): 1984 March 16; 15. PAGE COUNT: 258	
16. SUPPLEMENTARY NOTATION	Approved for public release: IAW AFR 190-17 LYNN E. WOLAVER 7 MAY 84 Dean for Research and Professional Development Air Force Institute of Technology (AU) Wright-Patterson AFB OH 45433

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	
12	01		Goal programming, linear programming, nuclear warfare, nuclear exchange model
15	06		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Title: BRIK: AN INTERACTIVE, GOAL PROGRAMMING MODEL FOR NUCLEAR EXCHANGE PROBLEMS

Thesis Chairmen: Ivy D. Cook, Jr., Lt Col, USAF
James K. Feldman, Maj, USAF

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ OTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL: Ivy D. Cook, Jr., Lt Col, USAF	22b. TELEPHONE NUMBER (Include Area Code): (513)255-3362; 22c. OFFICE SYMBOL: AFIT/ENS

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

SECURITY CLASSIFICATION OF THIS PAGE

19. Abstract:

BRIK is a unique nuclear exchange model (NEM). It is the first NEM to be both totally interactive and to rely entirely upon preemptive goal programming to drive its allocations. The program is written in FORTRAN 77 and exists on the VAX 11-780, using the UNIX operating system. It includes an internal allocation subroutine and is therefore completely transportable.

BRIK will allocate weapons to targets in order to meet damage expectancy (DE) goals of the user. The decision variable is a non-integer geometric mean representing the number of weapons allocated to an entire target class, using any of three objective functions and up to thirteen different allocation rules. The program is useful for a wide range of scenarios, including meeting DE goals with the available arsenal or creating an arsenal to meet established goals.

The programmed model uses a preemptive linear goal programming algorithm, allowing explicit preferential treatment of target classes, which can be placed into any of seven priorities. Target values can be input to differentiate target classes within the same priority. BRIK computes single-shot probability of survival using the nuclear targeting vulnerability (VN) system of the Defense Intelligence Agency or from vulnerability expressed as a PSI overpressure number. As currently dimensioned, the program can handle up to 20 target classes, 20 weapon classes, and 20 hedging constraints.

The report describes the physical and mathematical model, scenario limitations, input requirements, and the damage function. It inclues a user guide, variable and subroutine definitions, and a computer listing.